PR #34095 完整报告

sgl-project/sglang

config: the runner and scheduler read resolved config from the bags

合并时间：2026-08-10 05:45

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34095

执行摘要

- 一句话：runner 与 scheduler 配置读取迁移至配置袋，补齐 override 语义
- 推荐动作：值得精读。核心价值不在 diff 本身，而在两点：一是 is_ep_joiner / is_ep_scale_joiner 从 property 变函数的设计——因为 ep_join_mode 不是存档 leaf，无法走常规 leaf 读取路径，这是对配置系统 " 读什么、从哪读 " 边界的清晰注解；二是 review 中反复出现的 " 读 stamp 与读 bag 之争 " 与 " 单一快照 vs 多次读取 " 的取舍，直接展示了配置收敛迁移中最容易踩的坑。对配置系统演进方向感兴趣的工程师应重点阅读 runtime_context.py 的谓词实现、load_kv_cache_scales 的必传参数设计，以及作者 " 纯粹 read-source flip 不改变行为 " 的变更纪律。

功能与动机

PR body 明确指出动机："Where the process owns the object — the runner, the scheduler, the prefill bootstrap, the request receiver — reading the instance means missing every post-publish override, so those reads move to the namespaces"。即读取实例字段会丢掉发布后的覆盖配置，导致同一进程内不同组件对同一配置项看到不一致的值。同时作者划定了 " 刻意不翻转 " 的边界：tokenizer-manager 家族、entrypoints、MMEncoder、GrammarManager 和 nixl connector 是被调用方递交给配置的对象，多个 Engine 可共享一个进程，bags 是 last-publish-wins，因此它们必须继续读实例（"each is handed a config it must keep reading"）。

实现拆解

实现按 " 公共设施 → 各进程自有对象 → 测试配套 " 展开，共 5 步：

1. 新增弹性 EP 谓词（公共设施）：在 python/sglang/srt/runtime_context.py 新增 is_ep_joiner() 与 is_ep_scale_joiner() 两个模块级函数，直接读取已发布的 exec.moe.ep_join_mode leaf（"scale" / "recover"）。这两个成员在 ServerArgs 上是基于 ep_join_mode 的 property 而非实际存档的 leaf，因此无法通过 _configured_parallel 这类 leaf 读取路径拿到，必须新增函数形式，使其能跟随 post-publish override。
2. runner 侧配置读取迁移：python/sglang/srt/model_executor/model_runner.py 中 9 处 self.server_args.is_ep_scale_joiner / is_ep_joiner（_initialize_elastic_ep_joiner、maybe_init_expert_location_metadata、dist_barrier_after_load、_report_elastic_scale_failure、_finalize_scale_up、maybe_join_ep_ranks 等）全部替换为 runtime_context 谓词。configure_kv_cache_dtype 的请求 dtype 基线从

self.server_args.kv_cache_dtype 改为 get_model().kv_cache_dtype（bag 读取）。load_model 中调用 load_kv_cache_scales 时显式传入 get_model().kv_cache_dtype——因为 load_model 先于 configure_kv_cache_dtype 执行，此时 self.kv_cache_dtype 尚未写入，不能读 runner 自身 stamp。

3. 加载工具函数的数据契约收紧：python/sglang/srt/model_executor/model_runner_components/load_model_utils.py 的 load_kv_cache_scales 签名从 (*, model, server_args) 改为 (*, model, server_args, kv_cache_dtype: str)，且 kv_cache_dtype 为必传参数。docstring 说明原因：若设为带默认值回退到 server_args，未来任何调用方漏传都会变成隐藏的全局读取。FP8 scale 加载门控（fp8_e4m3）与 KV pool 配置从此读同一个 leaf，override 后二者不会分歧。
4. scheduler 与预填充侧迁移：python/sglang/srt/managers/scheduler.py 将 skip_tokenizer_init 改为构造时从 get_serving() 快照一次（self.skip_tokenizer_init），init_ipc_channels 与 init_tokenizer 都消费同一快照，避免 IPC 设置与 tokenizer 初始化在 override 后分歧；init_deterministic_inference_config 的 attention backend 键控改读 get_exec().kernel.attention_backend。python/sglang/srt/disaggregation/prefill.py 中 staging 校验、send_kv_chunk 的网格计算、CP guard 分别统一到 get_schedule().chunked_prefill_size 与 get_parallel().enable_prefill_context_parallel，消除 " 校验读实例、发送读 bag" 的分裂。request_receiver.py 的 _broadcast_reqs_across_ranks 改读 is_ep_scale_joiner()。
5. 投机解码与测试配套：python/sglang/srt/speculative/frozen_kv_mtp_worker_v2.py 的 _resolve_draft_backend_type 改为 get_spec().speculative_draft_attention_backend or attention_backends()[1]（读 bag 但刻意不读 runner stamp，理由见讨论）；dspark_planner.py 的 all-gather 谓词补上 get_parallel().pp_size、verify-lens 广播组改用 get_parallel().tp_size。测试侧：test_fp4_kv_cache_quant_method.py 从给 object.__new__(ModelRunner) 塞 server_args.kv_cache_dtype 改为通过 get_context().override_server_args(...) 发布配置；test_scheduler_init_req_max_new_tokens.py 的 setUp 从发布 dummy ServerArgs 改为 get_parallel().override(attn_dcp_size=1) 直接声明拓扑（生产代码读的是 live attn_dcp_size，原发布根本不控制断言乘数）。

关键文件：

- python/sglang/srt/model_executor/model_runner.py（模块 模型执行；类别 source；类型 data-contract；符号 load_model, configure_kv_cache_dtype, _initialize_elastic_ep_joiner, maybe_init_expert_location_metadata）：变更最集中的文件：9 处弹性 EP 谓词替换，configure_kv_cache_dtype 与 load_kv_cache_scales 的 dtype 读取统一到 bag，是 " 读取实例会错过 override" 问题的核心修复现场。
- python/sglang/srt/runtime_context.py（模块 运行时配置；类别 source；类型 core-logic；符号 is_ep_joiner, is_ep_scale_joiner）：新增 is_ep_joiner() / is_ep_scale_joiner() 两个谓词函数，是整个弹性 EP 读取迁移的公共设施；因为 ServerArgs 上的同名属性是基于 ep_join_mode 的 property 而非存档 leaf，必须用函数形式读取。
- python/sglang/srt/model_executor/model_runner_components/load_model_utils.py（模块 模型加载；类别 source；类型 data-contract；符号 load_kv_cache_scales）：

load_kv_cache_scales 签名收紧为必传 kv_cache_dtype，防止未来调用方漏传而形成隐藏的全局读取，是数据契约层面的关键变更。

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 init_ipc_channels, init_tokenizer, init_deterministic_inference_config）：skip_tokenizer_init 三处读取收敛为构造期单一快照，init_deterministic_inference_config 的 backend 键控改读 bag，是 scheduler 侧一致性的核心改动。
- python/sglang/srt/disaggregation/prefill.py（模块 预填充；类别 source；类型 dependency-wiring；符号 send_kv_chunk, PrefillBootstrapQueue）：staging 校验与 send 网格的 chunked_prefill_size 统一到 get_schedule()，CP guard 改读 get_parallel()，消除了 " 校验读实例、发送读 bag" 的潜在分歧。
- python/sglang/srt/speculative/frozen_kv_mtp_worker_v2.py（模块 投机解码；类别 source；类型 dependency-wiring；符号 _resolve_draft_backend_type）：_resolve_draft_backend_type 是讨论最激烈的点：最终不读 runner stamp，保留 get_spec().speculative_draft_attention_backend or attention_backends()[1] 链，避免丢掉 spec 设置与塌缩 topk==1 的 hybrid 配置。
- python/sglang/srt/managers/scheduler_components/request_receiver.py（模块 请求接收；类别 source；类型 dependency-wiring；符号 _broadcast_reqs_across_ranks）：请求接收器的 rank 间广播控制谓词改读 is_ep_scale_joiner()，补上该文件的归属。
- python/sglang/srt/speculative/dspark_components/dspark_planner.py（模块 投机规划；类别 source；类型 core-logic；符号 maybe_all_gather, _schedule_verify_lens）：dspark planner 的 all-gather 谓词补上 get_parallel().pp_size，verify-lens 广播组改用 get_parallel().tp_size，是投机解码侧拓扑读取的收尾。
- test/registered/unit/managers/test_scheduler_init_req_max_new_tokens.py（模块 调度测试；类别 test；类型 test-coverage；符号 setUp）：测试 double 从 " 发布 dummy ServerArgs" 改为 "get_parallel().override(attn_dcp_size=1) 声明拓扑 "，因为生产代码读的是 live attn_dcp_size，旧发布根本不控制断言乘数。
- test/registered/unit/layers/quantization/test_fp4_kv_cache_quant_method.py（模块 量化测试；类别 test；类型 test-coverage；符号 test_model_runner_rejects_legacy_fp4_alias）：测试随新契约调整：configure_kv_cache_dtype 现在读 get_model().kv_cache_dtype，测试改用 get_context().override_server_args(...) 发布配置而非塞实例属性。

关键符号：is_ep_joiner, is_ep_scale_joiner, load_kv_cache_scales, _resolve_draft_backend_type, configure_kv_cache_dtype, _initialize_elastic_ep_joiner, maybe_init_expert_location_metadata, send_kv_chunk, init_ipc_channels, init_tokenizer, init_deterministic_inference_config

关键源码片段

python/sglang/srt/model_executor/model_runner.py

变更最集中的文件：9 处弹性 EP 谓词替换，configure_kv_cache_dtype 与 load_kv_cache_scales 的 dtype 读取统一到 bag，是 " 读取实例会错过 override" 问题的核心修复现场。

```
# model_runner.py —— load_model() 尾部（head 版本）。
# 关键点：FP8 scale 加载门控必须与 configure_kv_cache_dtype 读同一个 leaf
# （bag 中的 kv_cache_dtype），而不是 runner 的启动记录，否则 override 之后
# "FP8 门控 " 与 "KV pool 配置 " 会各看各的值。
# 注意此时不能读 self.kv_cache_dtype：load_model 先于
# configure_kv_cache_dtype 执行，属性尚未写入。
load_kv_cache_scales(
    model=self.model,
    server_args=self.server_args,
    kv_cache_dtype=get_model().kv_cache_dtype,
)

self.sliding_window_size = resolve_sliding_window_size(
    self.model, self.model_config
)

# ... 权重加载统计、online 量化上报、RoPE 缓存预留 ...

dist_barrier_after_load(
    elastic_ep_backend=get_exec().moe.elastic_ep_backend,
    tp_rank=self.ps.tp_rank,
    # 弹性 EP 判词跟随已发布的 ep_join_mode leaf，而非启动时快照。
    # 这样 post-publish override 对 barrier 行为同样生效。
    is_ep_joiner=is_ep_joiner(),
)
```

python/sglang/srt/runtime_context.py

新增 is_ep_joiner() / is_ep_scale_joiner() 两个谓词函数，是整个弹性 EP 读取迁移的公共设施；因为 ServerArgs 上的同名属性是基于 ep_join_mode 的 property 而非存档 leaf，必须用函数形式读取。

```
# runtime_context.py —— 弹性 EP 谓词的 bag 化读取入口。
# 原实现读 ServerArgs 实例属性（self.server_args.is_ep_joiner），
# 该属性是 ep_join_mode 上的 property，不是真正存档的 leaf，
# 配置发布之后不会跟随 override，因此迁移为模块级谓词函数。
# 两个函数都直接读已发布的 exec.moe.ep_join_mode，保证与
# get_exec() 的 leaf 读取路径一致。



def is_ep_joiner() -> bool:
    """True in a process launched as an elastic-EP joiner (scale or recover).

    A predicate over the published ``exec.moe.ep_join_mode`` leaf, so it
    follows a post-publish override; the same-named ``ServerArgs`` property
    is the pre-publish equivalent.
    """
    return get_exec().moe.ep_join_mode in ("scale", "recover")

def is_ep_scale_joiner() -> bool:
```

```
    """True in a process launched as an elastic-EP scale-up joiner."""
    return get_exec().moe.ep_join_mode == "scale"
```

python/sglang/srt/speculative/frozen_kv_mtp_worker_v2.py

_resolve_draft_backend_type 是讨论最激烈的点：最终不读 runner stamp，保留 get_spec().speculative_draft_attention_backend or attention_backends()[1] 链，避免丢掉 spec 设置与塌缩 topk==1 的 hybrid 配置。

```
# frozen_kv_mtp_worker_v2.py —— draft 注意后端的类型解析。
# 保持原有链，但改从 bags 读取：spec 里显式设置的
# --speculative-draft-attention-backend 优先，否则用配置的
# decode backend（attention_backends()[1]，其本身回退到 base）。
# 刻意不读 runner 的 stamp：此 worker 构造 runner 时不传
# draft_attention_backend，stamp 是普通 decode/base pair，
# 读它会把 spec 设置丢掉；而且把 runner 钉死到单一后端会
# 塌缩 topk == 1 路径（该路径跑在 runner 自身后端上）的
# hybrid prefill/decode 配置。


def _resolve_draft_backend_type(self) -> str:
    return (
        get_spec().speculative_draft_attention_backend
        or attention_backends()[1]
    )
```

评论区精华

Review 中最重要的交锋有四处：

1. Codex P1：configure_kv_cache_dtype 的发布契约。Bot 指出轻量 runner（object.__new 构造、未发布进程配置）调用 configure_kv_cache_dtype 会先抛 ValueError: Global server args is not set yet!，导致 test_model_runner_rejects_legacy_fp4_alias 独立运行时失败。作者最初反驳，随后承认："Right this time, and my earlier reply was checking the wrong place — the test's publish-seeding fix lived in a later unit … That makes this unit not stand on its own, which the stacked-PR rule forbids." 修复方式是把测试的 publish 种子移入引入该读取的同一 commit，保证每个栈式 PR 单元自洽。
2. FrozenKV draft backend 读 stamp 还是读 bag。早期实现尝试读 draft_model_runner.decode_attention_backend_str（runner stamp），被 reviewer 指出：FrozenKVMTPDraftWorker 构造 runner 时并不传 draft_attention_backend，stamp 是普通 decode/base pair，会丢弃 --speculative-draft-attention-backend 设置并导致 topk > 1 启动失败。作者最初改为把显式 backend 传给 runner 构造，最终在二次 review 中 revert，保持 main 的链并改从 bags 读。作者结论："Reading the runner's stamp would have been wrong twice over: this worker does not hand its runner a draft backend, so the stamp is the ordinary pair … and pinning the runner to one backend collapses a hybrid prefill/decode configuration on the topk == 1 path." 同时承认 topk == 1 下 hybrid prefill/decode 若启用会有形状变化，需要在 PR body 中标注而

非宣称 "already the same"。

3. skip_tokenizer_init 单一快照。reviewer 指出 init_ipc_channels 读 bag、init_tokenizer 读实例快照会造成分歧，作者修复为单一快照："One value per process-owned scheduler, so an override cannot make IPC setup and tokenizer init disagree."
4. 确定性推理 truncation 键控的既有缺口。reviewer 指出 init_deterministic_inference_config 只按 base backend 键控，prefill-only 配置会漏设 truncation_align_size。作者拒绝在本 PR 顺手修复："this PR is a pure read-source flip whose whole claim is that no behaviour changes. Switching to attention_backends()[0] would start setting truncation_align_size for configurations that do not get it today … deserves its own PR and its own test"，记录为 follow-up。

- configure_kv_cache_dtype 在未发布上下文抛错导致单测独立运行失败 (testing): 作者承认违反栈式 PR 自包含纪律（"That makes this unit not stand on its own, which the stacked-PR rule forbids"），把测试的 publish 种子移入引入该读取的同一 commit，并验证该单元及其后每个单元可独立通过。
- FrozenKV draft backend 应读 runner stamp 还是读 bag (design): revert stamp 方案，保留 main 的链并改从 bags 读取：get_spec().speculative_draft_attention_backend or attention_backends()[1]。作者在 commit message 中明确承认 topk == 1 hybrid 配置是形状变化而非 "already the same"。
- skip_tokenizer_init 的三处读取收敛为单一快照 (design): fix：self.skip_tokenizer_init = get_serving().skip_tokenizer_init 在构造时快照一次，init_ipc_channels 与 init_tokenizer 都消费该快照，附注释说明两个消费方 "must not be able to disagree"。
- prefill staging 校验与 send grid 的 bag/instance 分裂 (correctness): fix：校验与 send grid 统一读 get_schedule().chunked_prefill_size；相邻 CP guard 因实例绑定消失被 ruff 发现，同步改为 get_parallel().enable_prefill_context_parallel。
- 确定性推理 truncation 键控仅覆盖 base backend (design): 作者刻意保持现状：base-only 查找是 main 既有行为，本 PR 是纯 read-source flip，切换到 attention_backends()[0] 会改变确定性推理数值行为，应独立 PR 单独测试。已记录为 follow-up。
- 测试 double 从发布 dummy config 改为状态 override (testing): 改为 get_parallel().override(attn_dcp_size=1) 声明拓扑并命名 live accessor，不再发布 dummy 配置。

风险与影响

- 风险：
  1. 行为等效性声明的例外：FrozenKV topk == 1 路径若此前依赖 hybrid prefill/decode 双后端，本次 _resolve_draft_backend_type 改为读 attention_backends()[1]（配置的 decode backend）后，draft 侧将固定为 decode 后端；作者已在 PR body 承认这是形状变化。
  2. 新配置契约（configure_kv_cache_dtype / load_kv_cache_scales）：现在要求 get_model() 已发布 model 配置，任何轻量构造 runner 的调用方（如测试 object.__new__(ModelRunner)）若不发布配置会直接抛 ValueError，属于 API 语义变

更。

3. skip_tokenizer_init 快照语义：初始化期间的 mid-init override 不再影响 tokenizer 初始化，作者视为特性（单一快照保证 IPC 与 tokenizer init 一致），但若未来有人期望热覆盖该值会失效。
4. 测试双写方式变化：两个测试从 " 塞实例属性 " 改为 " 发布配置 / 状态 override"，其他测试若沿用旧模式（直接构造 runner 后设 server_args 字段）可能在新契约下失败。
5. 栈式 PR 合并顺序耦合：本 PR 与 #34096、#34133 同属一个系列，若系列中相邻 PR 未同步合入 main，回查 skip_tokenizer_init、attn_dcp_size 等读取源时可能出现中间态不一致。 - 影响：影响范围为 srt 核心路径：model_runner.py（弹性 EP、KV dtype、FP8 scale 加载）、scheduler.py（tokenizer 初始化、确定性推理）、disaggregation/prefill.py（staging 网格）、frozen_kv_mtp_worker_v2.py 与 dspark_planner.py（投机解码）、request_receiver.py（请求分发）。 self.server_args.X 直读从 164 处降到 122 处，剩余 122 处被明确归类为 " 实例边界 "（tokenizer-manager 家族 90 处、entrypoints 与 MM 处理器 17 处等）。对用户无直接可见行为变化（默认单发布路径声明等效），但为配置 bag 化迁移扫清了进程自有对象这一最大障碍，使 post-publish override 能真正覆盖 runner 与 scheduler 的决策点。团队层面，该 PR 确立了 " 读取源一致性 " 与 " 栈式 PR 自包含 " 两条纪律，后续新增配置读取时需遵循同一模式。 - 风险标记：核心路径重构，行为等效性存在例外边界，新配置契约（须发布 model 配置），栈式 PR 合并耦合

关联脉络

- PR #34096 config: the KV-cache configurator reads the bags: 同系列前驱：KV 缓存配置器与 allocation sizing 已迁移到 bags，本 PR 将 runner 的 kv_cache_dtype / FP8 scale 读取与之对齐，二者共同保证 override 后 KV pool 与 FP8 门控一致。
- PR #34133 config: derive the runner's DCP topology from its ParallelState: DCP 拓扑改为从 ParallelState 派生后，本 PR 中 scheduler 测试的 attn_dcp_size 读取源与 get_parallel().override 语义依赖该变更，属于同一配置收敛方向。
- PR #34097 docs(skill): record where config is read now that the seed is off limits: 系列收尾文档：记录 config 的读取位置与种子禁读规则，与本 PR 的读取源迁移配套，防止后续开发重新引入实例读取。