PR #34263 完整报告

sgl-project/sglang

config: the last runner-side instance reads read the bags

合并时间：2026-08-15 15:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34263

执行摘要

- 一句话：runner 侧最后 6 处 server_args 直读迁移到配置 bags
- 推荐动作：值得精读。该 PR 是“把派生配置值从启动记录安全迁到 bags”的范本：单 bag 派生值抽 *_of(cfg) helper 并让 ServerArgs 委托、跨 bag 派生值采用双定义 + 分层一致性测试、以及用 attention_backends() 配对索引替代 getattr 运行时名字读取，都是可复用的设计决策。建议结合 #34269 的 bag 契约测试与 #34264 的 attention backend 配对迁移一起阅读，能看到同一模式的演进脉络。

功能与动机

配置系统正在从“启动记录 server_args 直读”迁移到“已发布 bags + override 感知 accessor”：解析管线结束后，业务代码若仍直接读启动记录，post-publish override 将无法生效。PR body 逐条列出了 6 处读取各自存在的原因，其中 DraftBackendFactory 通过 getattr(self.server_args, backend_name) 按运行时计算的名字读取 split backend，再手动回退 base 字段，正是配置普查（config census）文档记录过的盲点。PR 的目标是把非 tokenizer-manager 族的 self.server_args.X 读取压缩到只剩文档化的 5 处（encode server 自身记录、nixl connector 的 rank 运算、GrammarManager 的 handed instance）。

实现拆解

实现分 5 步完成，整体遵循“先建共享 helper，再补 bag accessor，最后迁移消费点并清理死数据”的顺序：

1. 共享 *_of helper 落地（python/sglang/srt/arg_groups/overrides.py，+37 行）：新增 modelexpress_transport_of(cfg)，统一 modelexpress_config（JSON 字符串 / dict / None）的解析并默认 "nixl"；新增 remote_instance_transfer_engine_of(cfg, load_format)，先短路 start_seed_via_transfer_engine，再按 (load_format or cfg.load_format) == "remote_instance" 失败关闭，最后看 backend 是否为 transfer_engine 或 modelexpress + transport 组合。ServerArgs.modelexpress_transport 保留自己的实例缓存解析（_parsed_modelexpress_config），但 ServerArgs.remote_instance_weight_loader_use_transfer_engine 改为委托该 helper（server_args.py 的 16 行改动）。
2. runtime_context.py 新增两个 bag accessor（+47 行）：remote_instance_transfer_engine_enabled(load_format) 是单 bag（model 叶子）派生值，直接委托 remote_instance_transfer_engine_of(get_model(), load_format)；

pre_capture_activation_reserve_mb(gpu_mem) 是跨 disagg / schedule / exec.graph / spec 四个 bag 加并行规模的派生值，无法单点共享，因此采用 mamba 式的"双定义"——ServerArgs 保留 publish 前版本，runtime_context 提供 publish 后版本，由分层一致性测试钉死相等。

3. 消费点迁移：scheduler.py 的 process_input_requests 改读 get_mm().mm_feature_transport；base_spec_worker.py 的 _build_hicache_draft_plan 改读 get_memory().enable_hierarchical_cache；draft_utils.py 的 _create_backend 改用 attention_backends() 配对按名字索引（草稿 runner 自身 stamp 仍优先）；model_runner.py 的 maybe_init_remote_instance_transfer_engine 与 remote_instance_weight_transporter.py 的 maybe_register_and_publish_weight_info 改调 bag accessor；kv_pool_runtime.py 的 compute_post_capture_kv_resize 与 kv_cache_configurator.py 统一走 bag 版 pre_capture_activation_reserve_mb。
4. 删除死数据：RemoteInstanceWeightTransporter 去掉无人读取的 server_args dataclass 字段与构造 kwarg；DraftBackendFactory 去掉停放的 server_args 记录和构造参数，4 个调用点（eagle_worker_v2.py、multi_layer_eagle_worker_v2.py、两个 attention-unittest kit 的 runner）同步移除传参。
5. 测试配套：test/registered/unit/test_runtime_context.py 扩展 _FakeResolvedArgs（新增 model / disagg / schedule / exec.graph / parallel 叶子），新增 test_activation_reserve_matches_the_member（7 组配置 × 3 档 gpu_mem 子测试）与 test_remote_instance_transfer_engine_matches_the_member（3 backend × 3 transport × 2 load_format × 2 seed 标志 × 3 override 全组合）；test_gpu_feature_transport.py 的调度器 MM 传输测试从 SimpleNamespace 伪造 server_args 改为 _publish 发布真实 bag 后再断言。全栈验证：8 分区 CPU 测试 6758 例 / 332 bad，相对栈基 6752 / 332 零新增失败；GLM-4.7-Flash + Qwen3-Next GDN 端到端与基线 byte-identical。

关键文件：

- python/sglang/srt/runtime_context.py（模块 配置上下文；类别 source；类型 core-logic；符号 remote_instance_transfer_engine_enabled, pre_capture_activation_reserve_mb）：新增两个 bag accessor（remote_instance_transfer_engine_enabled 与 pre_capture_activation_reserve_mb），是本 PR 的核心出口：前者委托单 bag helper，后者实现跨 4 bag 的派生计算并成为 KV 池与配置器共用的 reserve 来源。
- python/sglang/srt/arg_groups/overrides.py（模块 覆盖注册表；类别 source；类型 core-logic；符号 modelexpress_transport_of, remote_instance_transfer_engine_of）：新增 modelexpress_transport_of 与 remote_instance_transfer_engine_of 两个共享 helper，成为 transfer-engine 门控的唯一事实来源：ServerArgs member 委托它，runtime_context accessor 也委托它。
- test/registered/unit/test_runtime_context.py（模块 配置测试；类别 test；类型 test-coverage；符号 test_activation_reserve_matches_the_member, test_remote_instance_transfer_engine_matches_the_member）：扩展 _FakeResolvedArgs 并新增两个分层一致性测试，钉死 ServerArgs member 与 bag accessor 在输入矩阵上完全相等——这是双定义模式能长期安全的前提。

- python/sglang/srt/speculative/draft_utils.py（模块 草稿后端；类别 source；类型 dependency-wiring；符号 DraftBackendFactory._create_backend）：DraftBackendFactory 不再持有 server_args：_create_backend 用 attention_backends() 配对索引替代 getattr 运行时名字读取，这是配置普查盲点的直接修复。
- python/sglang/srt/model_executor/model_runner_components/remote_instance_weight_transporter.py（模块 权重传输；类别 source；类型 data-contract；符号 RemoteInstanceWeightTransporter, maybe_register_and_publish_weight_info）：数据契约清理：删除无人读取的 server_args dataclass 字段与构造 kwarg，门控改调 bag accessor remote_instance_transfer_engine_enabled()。
- python/sglang/srt/server_args.py（模块 启动配置；类别 source；类型 core-logic；符号 ServerArgs.remote_instance_weight_loader_use_transfer_engine）：ServerArgs.remote_instance_weight_loader_use_transfer_engine 的 14 行手写体压缩为对 remote_instance_transfer_engine_of(self, load_format) 的一行委托，消除语义双份。
- python/sglang/srt/model_executor/model_runner.py（模块 模型运行；类别 source；类型 data-contract；符号 ModelRunner.init_remote_instance_weight_transporter, ModelRunner.maybe_init_remote_instance_transfer_engine）：RemoteInstanceWeightTransporter 构造点去掉 server_args kwarg，maybe_init_remote_instance_transfer_engine 改调 bag accessor 并保留 draft_load_format 覆盖。
- python/sglang/srt/speculative/base_spec_worker.py（模块 推测工作器；类别 source；类型 dependency-wiring；符号 BaseSpecWorker._build_hicache_draft_plan）：_build_hicache_draft_plan 的 enable_hierarchical_cache 直读改为 get_memory() 叶子读取，是 hicache 门控跟随 override 的关键一跳。
- python/sglang/srt/model_executor/model_runner_components/kv_pool_runtime.py（模块 缓存池；类别 source；类型 data-contract；符号 compute_post_capture_kv_resize）：compute_post_capture_kv_resize 从 server_args 版本切到 bag-backed pre_capture_activation_reserve_mb，修复两个 reserve 在 post-publish override 下分叉的问题。
- test/registered/unit/multimodal/test_gpu_feature_transport.py（模块 特性传输；类别 test；类型 test-coverage；符号 _publish）：调度器 MM 传输测试从 SimpleNamespace 伪造 server_args 改为 _publish 发布真实 bags，保证被测路径与生产一致。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 Scheduler.process_input_requests）：process_input_requests 的 mm_feature_transport 门控改读 get_mm() 叶子，是调度器侧最后一处实例直读迁移。
- python/sglang/srt/mem_cache/kv_cache_configurator.py（模块 缓存配置；类别 source；类型 core-logic）：配置器侧的 pre_capture_activation_reserve_mb 调用点确认与 KV 池共用 bag 版本，保证两个 reserve 同源。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测工作器；类别 source；类型 core-logic）：DraftBackendFactory 构造点移除 server_args 实参，属于 4 个调用点联动的收尾。

- python/sglang/srt/speculative/multi_layer_eagle_worker_v2.py（模块 推测工作器；类别 source；类型 core-logic）：多层 Eagle worker 同样移除 DraftBackendFactory 的 server_args 实参。
- python/sglang/test/kits/attention_unittest/runner_modes/speculative_draft_extend_runner.py（模块 测试工具；类别 test；类型 test-coverage）：attention-unittest kit 中 DraftBackendFactory 的调用点同步去参，是测试基建与源码契约保持一致的证明。
- python/sglang/test/kits/attention_unittest/runner_modes/speculative_draft_runner.py（模块 测试工具；类别 test；类型 test-coverage）：与 extend 版本对称的调用点去参，保证 kit 在两个模式下都能通过 set_server_args 发布后构建工厂。

关键符号：remote_instance_transfer_engine_enabled, pre_capture_activation_reserve_mb, modelexpress_transport_of, remote_instance_transfer_engine_of, DraftBackendFactory._create_backend, ServerArgs.remote_instance_weight_loader_use_transfer_engine, compute_post_capture_kv_resize, maybe_register_and_publish_weight_info, maybe_init_remote_instance_transfer_engine, test_activation_reserve_matches_the_member, test_remote_instance_transfer_engine_matches_the_member

关键源码片段

python/sglang/srt/arg_groups/overrides.py

新增 modelexpress_transport_of 与 remote_instance_transfer_engine_of 两个共享 helper，成为 transfer-engine 门控的唯一事实来源：ServerArgs member 委托它，runtime_context accessor 也委托它。

```
def modelexpress_transport_of(cfg: Any) -> str:
    """解析 config 形状对象请求的 modelexpress transport。

    modelexpress_config 是 JSON 字符串（或已解析的 dict），不是独立叶子；
    这里是 transfer-engine 门控与未来 bag 读取方的共享解析。
    ServerArgs.modelexpress_transport 保留自己的实例级缓存解析
    （_parsed_modelexpress_config），两侧规则一致但缓存归属 seed 侧。
    """
    raw = cfg.modelexpress_config
    if raw is None:
        parsed = {}
    elif isinstance(raw, str):
        parsed = json.loads(raw)
    else:
        parsed = raw
    return parsed.get("transport", "nixl")


def remote_instance_transfer_engine_of(cfg: Any, load_format: Any = None) -> bool:
    """远程实例权重加载是否走 transfer engine（publish 前/后共用）。
```

```
    load_format 优先于 config 内的值：草稿 runner 在
    --speculative-draft-load-format 下需要自己的 transfer engine 判定。
    所有输入都是 model 叶子，因此同时服务 ServerArgs member 与 bag accessor。
    """
    # seed 阶段显式要求走 transfer engine，直接短路
    if cfg.remote_instance_weight_loader_start_seed_via_transfer_engine:
        return True
    # 非 remote_instance 加载格式，失败关闭
    if (load_format or cfg.load_format) != "remote_instance":
        return False
    backend = cfg.remote_instance_weight_loader_backend
    # transfer_engine 后端直接放行；modelexpress 后端需进一步看 transport 字段
    return backend == "transfer_engine" or (
        backend == "modelexpress"
        and modelexpress_transport_of(cfg) == "transfer_engine"
    )
```

test/registered/unit/test_runtime_context.py

扩展 _FakeResolvedArgs 并新增两个分层一致性测试，钉死 ServerArgs member 与 bag accessor 在输入矩阵上完全相等——这是双定义模式能长期安全的前提。

```
  def test_activation_reserve_matches_the_member(self):
    # 验证跨 bag 派生值：publish 前（ServerArgs member）与 publish 后
    #  （runtime_context accessor）必须在整个输入矩阵上相等，
    # 否则解析管线前后的决策会分叉
    from types import SimpleNamespace

    from sglang.srt.runtime_context import pre_capture_activation_reserve_mb

    graph = SimpleNamespace(decode=SimpleNamespace(max_bs=64))
    cases = (
        dict(disaggregation_mode="null", chunked_prefill_size=8192),
        dict(disaggregation_mode="null", chunked_prefill_size=-1),
        dict(disaggregation_mode="null", chunked_prefill_size=-1, max_prefill_tokens=1024),
        dict(disaggregation_mode="decode", max_running_requests=32),
        dict(disaggregation_mode="decode", max_running_requests=None),
        # decode 分离 + 草稿 token 参与 activation 估算的分支
        dict(disaggregation_mode="decode", max_running_requests=None, speculative_num_draft_
        tokens=4),
        dict(disaggregation_mode="null", chunked_prefill_size=8192, tp_size=8, pp_size=2),
    )
    for case in cases:
        for gpu_mem in (None, 20 * 1024, 80 * 1024):
            with self.subTest(gpu_mem=gpu_mem, **case):
                args = _FakeResolvedArgs(cuda_graph_config=graph, **case)
                get_context().set_server_args(args)
                self.assertEqual(
                    ServerArgs.pre_capture_activation_reserve_mb(args, gpu_mem),
                    pre_capture_activation_reserve_mb(gpu_mem),
```

)
```

### python/sglang/srt/speculative/draft_utils.py

DraftBackendFactory 不再持有 server_args：_create_backend 用 attention_backends() 配对索引替代 getattr 运行时名字读取，这是配置普查盲点的直接修复。

```
class DraftBackendFactory:
    def __init__(
        self,
        draft_model_runner,
        topk: int,
        speculative_num_steps: int,
        seed_dsa_topk_from_draft_extend: bool = False,
    ):
        # 不再持有 server_args：后端选择改从已发布 bags 读取
        self.draft_model_runner = draft_model_runner
        self.topk = topk
        self.speculative_num_steps = speculative_num_steps
        self.seed_dsa_topk_from_draft_extend = seed_dsa_topk_from_draft_extend
        # 草稿 runner 自己的后端优先，而不是进程级配置
        self.draft_attn_backend = draft_model_runner.draft_attention_backend

    def _create_backend(
        self, backend_name: str, backend_map: dict, error_template: str
    ):
        # 旧实现用 getattr(self.server_args, backend_name) 按运行时名字取值，
        # 再手动回退 base 字段；attention_backends() 返回的 pair 已带
        # base 回退，这里按名字索引即得到相同语义，且跟随 post-publish override。
        # 工厂只会收到 prefill_attention_backend / decode_attention_backend 两个名字。
        prefill_backend, decode_backend = attention_backends()
        configured = (
            decode_backend
            if backend_name == "decode_attention_backend"
            else prefill_backend
        )
        backend_type = self.draft_attn_backend or configured

        if backend_type not in backend_map:
            raise ValueError(error_template.format(backend_type=backend_type))

        return backend_map[backend_type]()
```

评论区精华

该 PR 的 review 全部由作者自查完成（COMMENTED 状态），核心讨论围绕 3 个“diff 外问题”与 1 个文档 nit：

- compute_post_capture_kv_resize 双 reserve 分叉风险（已修复）：review body 指出 kv_pool_runtime.py 仍调 model_runner.server_args.pre_capture_activation_reserve_m
```

b(...)，而配置器已改 bag 版本，post-publish override 会让预 profile 松弛量与 capture 后 headroom 分叉。作者在 issue comment 中确认折叠进本 PR，两处现在共用 bag-backed 版本。

- modelexpress_transport_of 文档言过其实（已修复）：inline nit 指出 docstring 声称“ServerArgs 属性与 post-publish accessor 都走同一个 parse”，但 ServerArgs.modelexpress_transport 仍读实例缓存的 _parsed_modelexpress_config。作者随后改写 docstring，使其只声称“transfer-engine 门控的共享解析”，并保留 seed 侧缓存。
- 死字段 / 停放记录清理（已修复）：RemoteInstanceWeightTransporter 保留了无人读取的 server_args 字段与构造 kwarg，DraftBackendFactory 停放了一份不再查询的记录；本 PR 删除字段与参数，并同步更新 4 个调用点。
- 流程反思：作者坦承这 3 个 diff 外问题搁置一天的原因是 review sweep 只扫 inline comments、未读 review body 的 “Issues outside the diff” 章节——“那是 channel miss，不是 disagreement”，并承诺将该渠道纳入后续 sweep。
  - modelexpress_transport_of 文档字符串言过其实 (documentation): docstring 改写为“transfer-engine 门控与未来 bag 读取方的共享解析”，ServerArgs 保留 seed 侧实例缓存；二次 re-review 确认表述与实现一致。
  - compute_post_capture_kv_resize 的 reserve 与配置器分叉风险 (correctness): 作者将三个 diff 外问题全部折叠进本 PR：compute_post_capture_kv_resize 改为调用 bag-backed pre_capture_activation_reserve_mb，两处 reserve 同源。
  - RemoteInstanceWeightTransporter 死字段与 DraftBackendFactory 停放记录 (design): 字段、构造 kwarg、构造参数全部删除，4 个调用点（两个 eagle worker + 两个 attention-unittest kit）同步去参；后续 re-review 确认无残留。
  - review sweep 渠道遗漏的流程反思 (other): 作者承诺将 review body 的 diff 外章节纳入后续 sweep 范围，并在此后的 re-review 中逐条闭环。

风险与影响

- 风险：
  1. DraftBackendFactory._create_backend 语义依赖命名约定：新实现按 backend_name == "decode_attention_backend" 决定取 decode 还是 prefill，review 确认工厂只会被传入这两个名字；若未来新增第三个 backend 名，会静默落到 prefill 分支。
  2. pre_capture_activation_reserve_mb 双定义需长期同步：ServerArgs 与 runtime_context 各持一份实现，任何一方新增输入叶子而未同步测试矩阵，分层一致性测试都可能漏检。
  3. modelexpress_transport_of 每次调用重新 json.loads：相比 ServerArgs._parsed_modelexpress_config 的实例缓存多一次解析，但仅出现在转移引擎门控与测试路径，热路径影响可忽略。
  4. 语义变化面：所有迁移点现在跟随 post-publish override，若下游有代码隐性依赖启动记录不变量，行为会变；端到端 byte-identical 验证覆盖了 GLM-4.7-Flash 与 Qwen3-Next GDN 两条当前路径，可基本排除回归，但未来新 override 场景需注意。 -

影响：影响面覆盖调度器 MM 传输门控、推测解码草稿后端选择、远程实例权重传输引擎初始化、KV 池 resize 预留等核心路径，涉及 16 个文件（+202/-44）。行为上从“读启动记录”变为“读已发布 bags”，语义应与现有路径等价，且后随 post-publishoverride——这是后续 6 个栈成员的配置边界基础。非 tokenizer-manager 族self.server_args.X 读取由 11 处降至 5 处。无外部 API / 协议变更，对用户无感知；对团队而言，*_of helper + 双定义 + 分层测试成为后续配置迁移的可复用范式。 - 风险标记：核心路径变更，配置语义跟随 override，双定义需测试钉住，跨模块调用点联动

关联脉络

- PR #34269 config: state the bag contract as what resolution produced, and the skill rule that goes with it: 重写 bag 契约测试与 skill 规则，为本 PR 的 bag accessor 语义提供契约基础；PR body 提到的“bag-contract case 被栈重写”即指此 PR。
- PR #34267 config: pin the supplied-instance surface that a raw record would change: supplied-instance 暴露面 ratchet 是本 PR“非 tokenizer-manager 族保留 5 处读取”边界的钉测来源。
- PR #34264 config: decisions keyed on the attention backend read the configured pair: 同栈成员：attention backend 决策改读 prefill/decode 配置对，与 DraftBackendFactory 改用 attention_backends() 配对索引是同一模式的两次落地。
- PR #34265 config: a named entry point for the resolution pipeline, and the last dynamic config read: 解析管线命名入口定义了 post-publish 边界，本 PR 的 bag accessor 正是在该边界之后消费配置。
- PR #34819 config: the post-publish consumers of the supplied-instance surface read the bags: 7 成员栈的后续成员，把 post-publish 消费者全面迁移到 bags，与本 PR 共享 model_runner 等相关文件与迁移模式。