PR #33336 完整报告

sgl-project/sglang

config: keep runtime hicache and weight-version updates off ServerArgs

合并时间：2026-08-03 12:23

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33336

执行摘要

- 一句话：配置更新迁出 ServerArgs，按作用域分置存储
- 推荐动作：值得精读。两个亮点：一是「进程级 bags vs per-engine manager」双轨配置归属的设计，直接决定多引擎共享 tokenizer 进程场景下的正确性；二是源码级守卫测试（TestControlPlaneFieldsAreNotReadFromTheInstance）用枚举模块 + 正则扫描把「静默过期读回」这类无报错回归钉死，该模式可复制到其他配置迁移场景。建议连同 #33337（FPM 端点入 bags）与 #33338（清除最后进程级配置读取）一起阅读，纵览 5 步系列全貌。

功能与动机

PR body 明确指出问题根源：运行时控制面变更此前直接改写 ServerArgs 实例，目的是让 readback 显示新值，但两边作用域不同——调度器侧直写实例时 namespace 读者（get_memory()）看不到变更，tokenizer 侧若写进程级 bags 又会因「several Engines can share one tokenizer process」把一引擎的变更泄漏进另一引擎的读回。此外 GitHub 将 chained-base 系列视为 stack，阻塞 base retargeting 与合并路径，前身 #33240 只能关闭，本 PR 改为基于 main 重提，代码与其最终版一致。

实现拆解

1. 调度器侧：HiCache attach/detach 落入进程级 config bags。python/sglang/srt/managers/scheduler.py 的 attach_hicache_storage_wrapped 与 detach_hicache_storage_wrapped 把 self.server_args.override("scheduler.attach_hicache", ...) 换成 get_context().override(...)。因为 get_internal_state 本来就报告 get_context().resolved_server_args_dict()，这一改让 get_memory() 等 namespace 读者与 resolved-config 读回同时看到变更，而发布出去的 ServerArgs 实例保持启动原样。
2. Tokenizer 侧：per-engine 更新沉淀在 Manager 上。python/sglang/srt/managers/tokenizer_manager.py 新增模块级 _SERVER_ARGS_FIELDS（由 ServerArgs dataclass 字段名构成的白名单）与 _MANAGER_OWNED_FIELDS（model_path、served_model_name），__init__ 增加 self._config_updates 列表。四个新方法：record_config_updates(source, **fields) 拒绝未知字段（防止 overlay 出幻影配置项）后追加记录；config_value(name) 按「更新优先、启动配置兜底」读单字段；resolved_config_dict(base) 把序列化 ServerArgs 当基座叠加更新；_dump_config_snapshot() 为转储生成配置快照。_update_model_path_info 改为只记录 load_format（路径字段走 _MANAGER_OWNED_FIELDS），

update_weights_from_disk 的 load_format 默认值、_handle_batch_output/_handle_abort_req 的 meta_info["weight_version"] 全部切到 config_value。tokenizer_control_mixin.py 的 HiCache mirror attach/detach 与 _update_weight_version_if_provided 三处写入同步改为 record_config_updates。

3. 读回端点整体切换 overlay 读取。http_server.py：/server_info 用 resolved_config_dict(dataclasses.asdict(server_args)) 生成返回体，/model_info 的 weight_version 用 config_value（顺带删掉重复键），/hicache/storage-backend 用循环 config_value 读四个 HiCache 字段，/update_weight_version 改用 record_config_updates。grpc_bridge.py 的 get_model_info/get_server_info、engine.py 的 get_server_info 同样切换；serving_chat/serving_classify/realtime session 的模型名读回也有小改。
4. 崩溃转储增强与降级路径。_dump_data_to_file 与 dump_requests_before_crash 的 payload 新增 config_updates 与 resolved_config 两个字段；当 ServerArgs 含不可 pickle 对象（如 --custom-sigquit-handler 的 lambda）时，回退逻辑同时把 server_args 与 resolved_config 置 None，确保请求数据仍能落盘。
5. 测试与守卫。新增 test_tokenizer_config_updates.py（216 行，覆盖 overlay 语义、实例不被改写、双引擎隔离、detach 读回、未知字段拒绝、dump 降级，以及核心守卫类 TestControlPlaneFieldsAreNotReadFromTheInstance——正则扫描 7 个 readback 模块，禁止 8 个控制面字段直读 self.server_args）、test_scheduler_hicache_attach.py（attach/detach 后 namespace 读者与 readback 一致）；test_server_info.py 的桩从 SimpleNamespace 换成真实 TokenizerManager；test_forward_pass_metrics.py 改为发布真实配置（按 body 描述）；writer ratchet 从 26 降到 19。

关键文件：

- python/sglang/srt/managers/tokenizer_manager.py（模块 配置管理；类别 source；类型 core-logic；符号 record_config_updates, config_value, _dump_config_snapshot, resolved_config_dict）：核心源码文件：新增 per-engine 配置更新记录机制（record_config_updates/config_value/resolved_config_dict/_dump_config_snapshot），并将 weight_version、load_format 等读回切换为 overlay 读取，同时增强崩溃转储 payload。
- test/registered/unit/managers/test_tokenizer_config_updates.py（模块 配置管理；类别 test；类型 test-coverage；符号 TestTokenizerConfigUpdates, TestControlPlaneFieldsAreNotReadFromTheInstance, CONTROL_PLANE_FIELDS, READMEBACK_MODULES）：新增 216 行测试：覆盖 overlay 语义、ServerArgs 实例不被改写、双引擎隔离、detach 读回、未知字段拒绝、dump 降级，并内置源码级守卫测试禁止 readback 模块直读控制面字段。
- test/registered/unit/managers/test_scheduler_hicache_attach.py（模块 调度器；类别 test；类型 test-coverage；符号 TestSchedulerHiCacheAttach, test_attach_reaches_the_namespace_readers, test_detach_clears_the_backend_for_the_same_readers）：新增 76 行测试：验证 HiCache attach/detach 经 get_context().override 后 namespace 读者（get_memory）与 resolved-config readback 一致、ServerArgs 实例保持原样。

- python/sglang/srt/entrypoints/http_server.py（模块 入口端点；类别 source；类型 core-logic；符号 model_info, server_info, hicache_storage_backend_status, update_weight_version）：HTTP 读回端点集中切换：/server_info 用 resolved_config_dict、/model_info 与 /hicache/storage-backend 用 config_value、/update_weight_version 用 record_config_updates，是外部消费者直接面对的行为变更面。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 attach_hicache_storage_wrapped, detach_hicache_storage_wrapped）：调度器侧 HiCache attach/detach 的写入目标从 self.server_args.override 改为 get_context().override，让 namespace 读者与读回端点同时看到变更。
- python/sglang/srt/managers/tokenizer_control_mixin.py（模块 配置管理；类别 source；类型 core-logic；符号 attach_hicache_storage, detach_hicache_storage, _update_weight_version_if_provided）：tokenizer 侧 HiCache mirror attach/detach 与 weight version 更新从 server_args.override 改为 record_config_updates，是本 PR per-engine 语义的直接落点。
- python/sglang/srt/entrypoints/grpc_bridge.py（模块 入口端点；类别 source；类型 core-logic；符号 get_model_info, get_server_info）：gRPC 读回端点 get_model_info/get_server_info 同步切换为 config_value / resolved_config_dict，保持与 HTTP 端点行为一致。
- python/sglang/srt/entrypoints/engine.py（模块 入口端点；类别 source；类型 core-logic；符号 get_server_info）：Engine.get_server_info 改用 resolved_config_dict，保证 Python API 层读回与 HTTP/gRPC 一致。
- python/sglang/srt/runtime_context.py（模块 配置管理；类别 source；类型 core-logic；符号 resolved_server_args_dict）：resolved_server_args_dict 的 docstring 更新，明确进程级 bags 与 per-engine manager 是两套独立日志、刻意不合并。
- test/registered/unit/entrypoints/test_server_info.py（模块 入口端点；类别 test；类型 test-coverage；符号 TestServerInfoControlPlaneUpdates, _call_server_info_with）：测试桩从 SimpleNamespace 换成真实 TokenizerManager，使 overlay 用例不会与生产代码漂移，并新增 control-plane overlay 用例。
- test/registered/unit/test_server_args_writer_ratchet.py（模块 配置管理；类别 test；类型 test-coverage）：writer ratchet 从 26 降到 19，锁定剩余 ServerArgs 写入点数量，防止本 PR 的工作回退。

关键符号：record_config_updates, config_value, resolved_config_dict, _dump_config_snapshot, attach_hicache_storage_wrapped, detach_hicache_storage_wrapped, attach_hicache_storage, detach_hicache_storage, _update_weight_version_if_provided, _update_model_path_info, update_weights_from_disk, model_info, server_info, hicache_storage_backend_status, update_weight_version, get_server_info, get_model_info, _handle_batch_output, _handle_abort_req

关键源码片段

python/sglang/srt/managers/tokenizer_manager.py

核心源码文件：新增 per-engine 配置更新记录机制（record_config_updates/config_value/resolved_config_dict/_dump_config_snapshot），并将 weight_version、load_format 等读回切换为 overlay 读取，同时增强崩溃转储 payload。

```
# ServerArgs 的字段白名单：控制面更新只能落在这些已声明字段上
# （防止未知键在 /server_info 的 overlay 中变成"幻影配置项"）
_SERVER_ARGS_FIELDS = frozenset(f.name for f in dataclasses.fields(ServerArgs))

# 这两个字段由 Manager 直接持有（属性即事实来源），不走 overlay 列表
_MANAGER_OWNED_FIELDS = ("model_path", "served_model_name")


class TokenizerManager(TokenizerControlMixin, TokenizerManagerScoreMixin):
    def record_config_updates(self, source: str, **fields) -> None:
        """记录一次运行期控制面变更（per-engine 语义）。

        多个 Engine 可共享同一个 tokenizer 进程，所以不能写进程级
        config bags （会串到别的 Engine 的 readback）；变更先攒在
        本 Manager 上，由读回端点以 overlay 方式叠加展示。
        """
        unknown = sorted(f for f in fields if f not in _SERVER_ARGS_FIELDS)
        if unknown:
            raise ValueError(
                f"{unknown} are not ServerArgs fields; the readback endpoints "
                "overlay these onto a serialized ServerArgs, so an unknown key "
                "would surface as a phantom config entry."
            )
        self._config_updates.append((source, dict(fields)))

    def config_value(self, name: str):
        """读取单个字段的生效值：控制面更新优先，其次启动配置兜底。"""
        if name in _MANAGER_OWNED_FIELDS:
            return getattr(self, name)
        for _source, fields in reversed(self._config_updates):
            if name in fields:
                return fields[name]
        return getattr(self.server_args, name)

    def resolved_config_dict(self, base: Dict[str, Any]) -> Dict[str, Any]:
        """把序列化后的 ServerArgs 作为基底，叠加上本引擎的控制面更新。

        /server_info、gRPC get_server_info 等都以此生成返回体，
        保证读回反映当前生效配置而不是启动记录。
        """
        resolved = dict(base)
        for _source, fields in self._config_updates:
            resolved.update(fields)
        for name in _MANAGER_OWNED_FIELDS:
            resolved[name] = getattr(self, name)
```

```
        return resolved
```

test/registered/unit/managers/test_tokenizer_config_updates.py

新增 216 行测试：覆盖 overlay 语义、ServerArgs 实例不被改写、双引擎隔离、detach 读回、未知字段拒绝、dump 降级，并内置源码级守卫测试禁止 readback 模块直读控制面字段。

```
# 运行期会变化的控制面字段清单：这些字段一律禁止从 ServerArgs 实例直读
CONTROL_PLANE_FIELDS = (
    "weight_version",
    "model_path",
    "served_model_name",
    "load_format",
    "hicache_storage_backend",
    "hicache_storage_backend_extra_config",
    "hicache_storage_prefetch_policy",
    "hicache_write_policy",
)

# 豁免行：Prometheus label 集合必须与序列同生命周期固定，
# 因此 metrics collector 保留服务启动时的名字是刻意为之
EXEMPT_LINES = (
    (
        "srt/managers/tokenizer_manager.py",
        '"model_name": self.server_args.served_model_name',
    ),
)


class TestControlPlaneFieldsAreNotReadFromTheInstance(CustomTestCase):
    def test_readbacks_go_through_the_manager(self):
        # 扫遍所有读回模块的源码，抓到任何直读
        # self.server_args.< 控制面字段 > 的地方——失败模式是
        # 静默过期读回（不报错、只返回旧值），值得用守卫测试钉死
        root = Path(next(iter(sglang.__path__)))
        patterns = [
            re.compile(
                rf"self\.server_args\.{f}\b|tokenizer_manager\.server_args\.{f}\b"
            )
            for f in CONTROL_PLANE_FIELDS
        ]
        stale = []
        for rel in READBACK_MODULES:
            paths = (
                sorted((root / rel).rglob("*.py"))
                if (root / rel).is_dir()
                else [root / rel]
            )
            for path in paths:
                for number, line in enumerate(path.read_text().split("\n"), 1):
```

```
                if any(
                    rel_exempt == path.relative_to(root).as_posix()
                    and needle in line
                    for rel_exempt, needle in EXEMPT_LINES
                ):
                    continue
                if any(p.search(line) for p in patterns):
                    stale.append(
                        f"{path.relative_to(root)}:{number}: {line.strip()}"
                    )
        self.assertEqual(
            stale,
            [],
            "control-plane fields change at runtime and the update lives on the "
            "TokenizerManager; read them with config_value() / "
            "resolved_config_dict() so the readback reflects the change:\n"
            + "\n".join(stale),
        )
```

test/registered/unit/managers/test_scheduler_hicache_attach.py

新增 76 行测试：验证 HiCache attach/detach 经 get_context().override 后 namespace 读者（get_memory）与 resolved-config readback 一致、ServerArgs 实例保持原样。

```
class TestSchedulerHiCacheAttach(CustomTestCase):
    def _scheduler(self, **fields):
        # 测试专用：向 config tiers 发布一份携带指定字段的 ServerArgs
        # （install 替换当前槽位，restore 还原），再用 __new__ 绕过真实
        # 构造函数拼装出被测方法所需的最小 Scheduler 实例
        override = get_context().override_server_args(
            enable_hierarchical_cache=True, **fields
        )
        self.server_args = override.install()
        self.addCleanup(override.restore)

        scheduler = Scheduler.__new__(Scheduler)
        scheduler.server_args = self.server_args
        scheduler.enable_hierarchical_cache = True
        scheduler.enable_hicache_storage = False
        scheduler.is_fully_idle = lambda: True
        scheduler.tree_cache = SimpleNamespace(
            attach_storage_backend=lambda **kwargs: (True, "attached"),
            detach_storage_backend=lambda: (True, "detached"),
        )
        return scheduler

    def test_attach_reaches_the_namespace_readers(self):
        scheduler = self._scheduler(hicache_storage_backend=None)
        out = scheduler.attach_hicache_storage_wrapped(
            AttachHiCacheStorageReqInput(
```

```
            hicache_storage_backend="file",
            hicache_write_policy="write_through",
        )
    )
    self.assertTrue(out.success)
    # 关键断言：namespace 读者（get_memory）与 resolved-config readback
    # 都能看到后端已挂载，而发布出去的 ServerArgs 实例保持启动原样
    self.assertEqual(get_memory().hicache_storage_backend, "file")
    self.assertEqual(
        get_context().resolved_server_args_dict()["hicache_storage_backend"],
        "file",
    )
    self.assertIsNone(self.server_args.hicache_storage_backend)
```

评论区精华

本 PR 自身无 review 评论（review_comments_count: 0），讨论均发生在前身 #33240 上，PR body 明确「The review discussion and the triage of each round of comments is on #33240; the code here is identical to that PR's final revision」。核心要点：① GitHub 将 chained-base 系列视为 stack，阻塞 base retargeting 与除异步端点外的所有合并路径，因此弃用 #33240、基于 main 重提；② 配置归属边界——调度器侧写进程级 bags、tokenizer 侧留 per-engine manager，runtime_context.py docstring 明确「The two are separate logs, not one merged dict」，两套日志刻意不合一；③ 守卫测试必要性——直读 ServerArgs 的失败模式是静默过期读回，不报错只返回旧值，故用枚举模块 + 正则扫描钉死 7 个 readback 模块，Prometheus label 集合因须与序列同生命周期而豁免；④ 验证空白——HiCache attach/detach over HTTP 需带分层缓存的真实服务，作者明言未在本地端到端验证。Issue 上仅有一条 Gemini Code Assist 停服的机器人通知，无实质内容。

- chained-base 系列被 GitHub 视为 stack 导致合并阻塞 (other): 弃用 #33240，本 PR 基于 main 重提，代码与其最终版一致。
- 进程级 bags 与 per-engine manager 的配置归属边界 (design): 两套日志刻意不合一，防止多 Engine 共享 tokenizer 进程时串扰。
- 守卫测试防止静默过期读回回潮 (testing): 新增 TestControlPlaneFieldsAreNotReadFromTheInstance，扫描 7 个 readback 模块中的 8 个控制面字段。
- HiCache attach/detach HTTP 路径未端到端验证 (testing): 依赖单元测试覆盖逻辑路径，端到端验证留待后续；已标注为已知验证空白。

风险与影响

- 风险：
  - 静默过期读回（主要回归面）：外部消费者（监控工具、KV-aware 路由、依赖 /get_server_info 的旧客户端）依赖读回字段的即时性。守卫测试只覆盖 READBACK_MODULES 内 7 个模块，未来新增读回模块若直读 self.server_args.<控制面字段>，不会被现行正则捕获，EXEMPT_LINES 也会留出口子。

- 多 Engine 并发叠加：_config_updates 是单一列表，同进程多 Engine 的更新按追加顺序后写覆盖；若两引擎并发更新同一字段，读出为最后写入者，符合 per-manager 语义但需业务侧确认可接受。
- 崩溃转储降级：_dump_config_snapshot 遇不可序列化配置返回 None，转储会缺失 resolved config 但保留请求数据；回退同时置空两个字段的行为已有测试覆盖。
- 端到端验证空白：HiCache HTTP attach/detach 路径未在本地验证，属于已知风险，依赖单测覆盖逻辑路径。
- 影响：影响范围集中在控制面与可观测性层：所有 introspection 端点（HTTP /server_info、/model_info、/hicache/storage-backend，gRPC get_model_info/get_server_info，Engine.get_server_info）返回的 weight_version、model_path、HiCache 字段现在反映运行期实际生效配置；对共享 tokenizer 进程的多引擎部署是正确性修复（此前进程级 bags 下直写实例会串扰）；对外部消费者字段兼容性保持，仅语义变为「当前生效配置」。团队层面确立了「进程级 bags vs per-engine manager」的配置分层边界，为系列后续（#33338 清除全部进程级配置字段直读）铺路。
- 风险标记：readback 面广、漏改即静默过期，守卫测试依赖正则扫描存在盲区，HiCache attach HTTP 路径未端到端验证，崩溃转储配置快照可能缺失

关联脉络

- PR #33240 同系列前身 PR（chained-base 版，已关闭未合并；标题未提供）：PR body 明确本 PR 是其替代实现，代码与其最终版一致；因 GitHub stack 机制阻塞合并而关闭，全部 review 讨论都在该 PR 上。
- PR #33337 observability: publish the generated forward-pass-metrics endpoint to the bags: 同为 5 步配置梳理系列前序步骤：FPM 端点写入迁移至配置 bags，与本 PR 共享 runtime_context.override 机制与 ratchet 测试，定义了调度器侧 ' 写 bags' 的模式。
- PR #33338 config: retire the last process-global config field reads: 系列后续步骤：清除最后一批进程级配置字段直读并迁移至命名空间访问器，与本 PR 一脉相承，共同把 ServerArgs 收敛为 ' 启动记录 ' 语义。
- PR #33294 test: stand up the config tiers two unit tests read from: 发布真实配置修复两个 main 分支失败的单测，与 config tiers 系列同属配置分层工作，为本 PR 的测试桩升级（真实 TokenizerManager 替代 SimpleNamespace）提供了方向。