PR #36300 完整报告

sgl-project/sglang

config: the model-config cache keys on the path the record carried

合并时间：2026-08-25 18:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36300

执行摘要

- 一句话：修复对象存储路径下模型配置缓存误失效导致启动崩溃
- 推荐动作：值得精读，特别是面向维护启动配置与参数解析的工程师。核心看点：(1) 如何区分 " 记录路径移动 " 与 " 配置自身路径移动 "，确定缓存失效键；(2) 只读保护如何通过 _CACHE_SLOTS 与缓存槽共存而不破坏 " 解析后配置不可变 " 的边界；(3) 测试如何用真实 ModelConfig 路径 + stub 下载锁定回归。

功能与动机

PR body 指出 test/registered/model_loading/test_runai_model_loader.py 是 2026-08-23 nightly 上 nightly-1-gpu-large 唯一失败的用例：所有 --load-format runai_streamer 启动都会在 scheduler 中崩溃，报错 server_args.model_config assigned after resolution; server_args is read-only。根因是 #35907 让 get_model_config() 在缓存对象报告的 model_path 与记录不一致时丢弃缓存，却把两种无关的路径移动合并了：resolution 移动记录的路径（GGUF/ModelScope 需要重建）与 ModelConfig 在构造函数内自移路径指向本地拉取目录（对象存储场景仍描述同一 checkpoint，不应重建）。

实现拆解

1. 缓存键语义调整（python/sglang/srt/server_args.py::get_model_config）：新增私有属性 _model_config_built_from 记录填充缓存时的 self.model_path，命中判断从 " 缓存对象 model_path 是否等于记录路径 " 改为 " 记录自身路径自填充后是否移动 "：built_from is None or built_from == self.model_path 时直接返回缓存。built_from is None 兼容 fixture 注入的配置；重建时同时写回缓存对象和键。
2. 只读保护放行缓存槽（python/sglang/srt/server_args.py::__setattr__与 _CACHE_SLOTS）：原保护按拼写分类，公共名称视为已解析配置，model_config 因此被误拦截。新增 _CACHE_SLOTS = frozenset({"model_config"})，在保护条件中加入 name not in _CACHE_SLOTS，使缓存槽在已解析记录上可写；下划线键（如 _model_config_built_from）本就豁免。同时修正 _underscore_field_names() 的 docstring。
3. 回归测试（新增test/registered/unit/server_args/test_model_config_cache.py，171行）：注册 CPU CI（register_cpu_ci(est_time=10, suite="base-a-test-cpu")），用真实 ModelConfig、is_runai_obj_uri、_maybe_pull_model_for_runai 路径，仅 stub 下载。四个用例覆盖对象存储保持缓存、GGUF/ModelScope 重建（#35907 失效语义不变）、已解

析记录缓存 refill、fixture 配置直通，并处理 EnvField 与环境变量恢复。

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 get_model_config, setattr, _CACHE_SLOTS, _underscore_field_names）：核心修复文件：get_model_config() 的缓存键改为记录填充时携带的路径（_model_config_built_from），并新增 _CACHE_SLOTS 让 __setattr__只读保护放行缓存槽写入，解决对象存储 URI 模型启动崩溃。
- test/registered/unit/server_args/test_model_config_cache.py（模块 配置缓存；类别 test；类型 test-coverage；符号 TestTheModelConfigCache, test_a_configuration_that_repoints_itself_stays_cached, test_a_declared_model_path_rebuilds_the_configuration）：新增的回归测试文件：用真实 ModelConfig 与对象存储拉取路径（仅 stub 下载）覆盖缓存键语义的四种形状，其中两个用例在未打补丁的 main 上会因 AttributeError 失败。

关键符号：get_model_config, setattr, _underscore_field_names, test_a_configuration_that_repoints_itself_stays_cached, test_a_declared_model_path_rebuilds_the_configuration

关键源码片段

python/sglang/srt/server_args.py

核心修复文件：get_model_config() 的缓存键改为记录填充时携带的路径（_model_config_built_from），并新增 _CACHE_SLOTS 让 __setattr__只读保护放行缓存槽写入，解决对象存储 URI 模型启动崩溃。

```
# python/sglang/srt/server_args.py —— get_model_config 缓存键语义
def get_model_config(self):
    # 延迟导入以避免循环依赖
    from sglang.srt.configs.model_config import ModelConfig

    memo = getattr(self, "model_config", None)
    if memo is not None:
        # 缓存键是填充缓存时记录携带的路径（_model_config_built_from）。
        # GGUF / ModelScope 处理器会声明不同的 model_path，配置构建于其前
        # 则描述另一份 checkpoint，需要重建；而对象存储 URI 场景下是
        # ModelConfig 自己在构造函数里把 model_path 指向本地拉取目录，
        # 记录的路径并未移动，缓存仍然有效。fixture 注入的配置没有键，直接返回。
        built_from = getattr(self, "_model_config_built_from", None)
        if built_from is None or built_from == self.model_path:
            return memo

    model_config = ModelConfig.from_server_args(self)
    self.model_config = model_config
    self._model_config_built_from = self.model_path
    if model_config.is_hybrid_swa:
        logger.info(
```

```
            "Hybrid SWA model detected. architectures=%s",
            model_config.hf_config.architectures,
        )
    return model_config


# 只读保护放行的缓存槽：记录从自身派生的值不属于已解析配置，
# 而可能使缓存失效的键必须能在已解析记录上写回。
_CACHE_SLOTS = frozenset({"model_config"})


def __setattr__(self, name, value):
    # 已物化后，字段是最终启动配置，默认只读；唯一例外是下划线字段
    #（记录自身簿记）以及 _CACHE_SLOTS 中列出的缓存槽。
    if (
        getattr(self, "_declarations_materialized", False)
        and not getattr(self, "_internal_write", False)
        and name not in _CACHE_SLOTS
        and (not name.startswith("_") or name in _underscore_field_names())
    ):
        raise AttributeError(
            f"server_args.{name} assigned after resolution; server_args is "
            "read-only -- use get_context().override(source, ...) to change "
            "resolved config; a value one runner owns travels as a "
            "constructor argument."
        )
    object.__setattr__(self, name, value)
```

test/registered/unit/server_args/test_model_config_cache.py

新增的回归测试文件：用真实 ModelConfig 与对象存储拉取路径（仅 stub 下载）覆盖缓存键语义的四种形状，其中两个用例在未打补丁的 main 上会因 AttributeError 失败。

```
# test/registered/unit/server_args/test_model_config_cache.py —— 回归测试
class TestTheModelConfigCache(CustomTestCase):
    def setUp(self):
        # 解析过程会写环境变量并翻转 EnvField 的描述符标志；
        # os.environ 不携带后者，因此需要同时保存并恢复。
        fields = {}
        for klass in reversed(type(envs).__mro__):
            for name, field in vars(klass).items():
                if isinstance(field, EnvField):
                    fields[name] = field
        state = (
            dict(os.environ),
            {name: field._set_to_none for name, field in fields.items()},
        )
        self.addCleanup(self._restore, state)

    @staticmethod
```

```
def _restore(state):
    saved_environ, saved_none_flags = state
    os.environ.clear()
    os.environ.update(saved_environ)
    for name, was_none in saved_none_flags.items():
        getattr(type(envs), name)._set_to_none = was_none

def test_a_configuration_that_repoints_itself_stays_cached(self):
    # 对象存储形状：ModelConfig 自身把 model_path 指向本地拉取目录，
    # 记录仍保留 URI，缓存应保持命中。
    pulled = self._checkpoint()
    self._pulled_to(pulled)

    server_args = self._resolved(
        model_path=_OBJECT_STORE_URI, load_format="runai_streamer"
    )
    cached = server_args.__dict__["model_config"]
    self.assertIsInstance(cached, ModelConfig)
    self.assertEqual(server_args.model_path, _OBJECT_STORE_URI)
    self.assertEqual(cached.model_path, pulled)
    self.assertIs(server_args.get_model_config(), cached)
```

评论区精华

PR 没有 review 评论；核心讨论体现在 PR body 与提交信息中。作者把两种路径移动区分开：resolution 会移动记录的路径（GGUF/ModelScope 场景，应重建），而 ModelConfig 在对象存储场景下自移 model_path（不应重建）。据此把缓存键定为 " 记录填充缓存时携带的路径 " 而非缓存对象自身的 model_path，既保留 #35907 的失效语义，又解决对象存储误重建。没有遗留的未解决疑虑。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险：失效条件改为 built_from != self.model_path 后，若未来有逻辑在构造配置前后修改 self.model_path 且不希望重建，可能被误判失效；测试覆盖了当前已知的 GGUF/ModelScope 与对象存储两条路径。
  2. 只读保护变宽：_CACHE_SLOTS 使 model_config 成为已解析记录上唯一可写的公共属性，其他代码若误写该属性不再被拦截（此前会抛 AttributeError）。这是有意豁免，但扩大了可变面。
  3. 影响范围：仅影响启动期配置构建，is_hybrid_swa 等决策读取新构建的配置，行为不变；不涉及推理路径，无性能与精度影响。
  4. 测试依赖：新测试 stub 了对象存储下载，注册为 CPU CI，若 runai_utils 接口变化需同步更新。 - 影响：对用户：所有 --load-format runai_streamer 以及远程连接器（gs://、s3://、az://）的模型启动从崩溃恢复，Nightly CI 的 nightly-1-gpu-large 回归转

绿。对系统：get_model_config() 在对象存储路径下不再重复重建 ModelConfig，减少启动期冗余解析；缓存失效语义更精确。对团队：新增的 4 个单元测试明确了缓存键边界，为后续 server_args 解析 / 只读机制重构提供回归保护。 - 风险标记：对象存储启动路径，缓存键语义调整，只读保护豁免，新增回归测试

关联脉络

- PR #36178 [Fix] Keep the MiniCPM-SALA config reads visible to the resolution ratchets: 同属 server_args 解析 / 只读机制域：36178 调整 overrides.py 与解析棘轮的可见性，本 PR 进一步调整 __setattr__只读保护的豁免范围（_CACHE_SLOTS），两者都在维护解析后配置的边界语义。
- PR #36235 [CI] Stop the resolution ratchets re-parsing the whole package: 与本 PR 同处 test/registered/unit/server_args/ 测试族，关注解析后配置的读取路径；本 PR 新增的 test_model_config_cache.py 与该文件族共同守护 server_args 解析行为。
- PR #36240 [CI] Stop the config ratchets re-parsing the package on every scan: 同样优化 server_args 配置解析相关注册测试的 CI 耗时，与本 PR 新增测试直接同目录，属于同一测试基础设施演进线。