PR #31211 完整报告

sgl-project/sglang

Fix processor config loading for object-storage model paths

合并时间：2026-07-15 15:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31211

执行摘要

- 一句话：修复 object-storage 模型路径的 processor 加载
- 推荐动作：值得快速合入，变更微小且逻辑清晰。推荐了解 object-storage 缓存机制的读者留意 resolve_runai_obj_uri 的实现；其他读者可直接合并。

功能与动机

PR body 指出，get_processor() 已对 tokenizer_name 调用 resolve_runai_obj_uri()，但当 model_name 单独指定时，AutoConfig.from_pretrained(model_name) 会使用未解析的 URI，绕过为 RunAI 后端（S3/GCS/Azure）准备好的本地缓存，导致 processor 初始化失败。

实现拆解

1. 源码变更：在 python/sglang/srt/utils/hf_transformers/processor.py 的 get_processor() 函数中，第 156-157 行新增对 model_name 的 URI 解析：if model_name is not None: model_name = resolve_runai_obj_uri(model_name)。这样后续 AutoConfig.from_pretrained(model_name) 使用本地缓存路径而非原始远程 URI。
2. 测试新增：在 test/registered/unit/utils/test_hf_transformers.py 中新增 TestGetProcessor 测试类和 test_resolves_model_name_before_loading_config 方法，模拟 tokenizer 为本地路径、model_name 为 s3://bucket/model 的场景，通过 mock 验证 resolve_runai_obj_uri 被正确调用，并且 AutoConfig.from_pretrained 接收的是解析后的本地路径。
3. 配套调整：测试文件中新增导入 sglang.srt.utils.hf_transformers.processor as processor_utils 和 MagicMock，利用 patch.multiple 完成 mock。

关键文件：

- python/sglang/srt/utils/hf_transformers/processor.py（模块 工具函数；类别 source；类型 core-logic）：核心修复：在 get_processor() 中对 model_name 调用 resolve_runai_obj_uri，确保 AutoConfig 使用本地缓存路径。仅 +2 行。
- test/registered/unit/utils/test_hf_transformers.py（模块 测试；类别 test；类型 test-coverage；符号 TestGetProcessor, test_resolves_model_name_before_loading_config, resolve_uri）：新增 CPU 回归测试，模拟 local tokenizer + remote model_name 场景，验证修复正确性。

关键符号：get_processor, resolve_runai_obj_uri, test_resolves_model_name_before_loading_config

关键源码片段

python/sglang/srt/utils/hf_transformers/processor.py

核心修复：在 get_processor() 中对 model_name 调用 resolve_runai_obj_uri，确保 AutoConfig 使用本地缓存路径。仅 +2 行。

```
# python/sglang/srt/utils/hf_transformers/processor.py (lines 138-178)

def get_processor(
    tokenizer_name: str,
    *args,
    tokenizer_mode: str = "auto",
    trust_remote_code: bool = False,
    tokenizer_revision: Optional[str] = None,
    use_fast: Optional[bool] = True,
    tokenizer_backend: str = "huggingface",
    model_name: Optional[str] = None,
    **kwargs,
):
    if tokenizer_backend == "fastokens":
        from .tokenizer import _ensure_fastokens_patched
        _ensure_fastokens_patched()

    revision = kwargs.pop("revision", tokenizer_revision)
    # tokenizer 路径已在处理
    tokenizer_name = resolve_runai_obj_uri(tokenizer_name)

    # 新增：对 model_name 做同样处理，确保后续 AutoConfig 使用本地缓存路径
    if model_name is not None:
        model_name = resolve_runai_obj_uri(model_name)

    if is_mistral_model(tokenizer_name):
        config = load_mistral_config(...)
    elif model_name is not None:
        # 这里现在接收本地路径，而非原始 S3/GCS/Azure URI
        config = AutoConfig.from_pretrained(
            model_name,
            trust_remote_code=trust_remote_code,
            revision=revision,
            **kwargs,
        )
    else:
        config = AutoConfig.from_pretrained(
            tokenizer_name,
            trust_remote_code=trust_remote_code,
            revision=revision,
```

```
            **kwargs,
        )
    # 后续处理不变 ...
```

test/registered/unit/utils/test_hf_transformers.py

新增 CPU 回归测试，模拟 local tokenizer + remote model_name 场景，验证修复正确性。

```
# test/registered/unit/utils/test_hf_transformers.py ( 新增 get_processor 小节 )

import sglang.srt.utils.hf_transformers.processor as processor_utils
from unittest.mock import MagicMock, patch

class TestGetProcessor(unittest.TestCase):
    def test_resolves_model_name_before_loading_config(self):
        # 模拟：tokenizer 本地，model_name 为远程 S3 路径
        remote_model = "s3://bucket/model"
        local_model = "/cache/model"
        config = SimpleNamespace(model_type="clip", auto_map={})
        loaded_processor = MagicMock()
        loaded_processor.tokenizer.chat_template = "template"
        auto_config = MagicMock()
        auto_config.from_pretrained.return_value = config
        auto_processor = MagicMock()
        auto_processor.from_pretrained.return_value = loaded_processor

        def resolve_uri(path):
            return local_model if path == remote_model else path

        with patch.multiple(
            processor_utils,
            resolve_runai_obj_uri=MagicMock(side_effect=resolve_uri),
            AutoConfig=auto_config,
            AutoProcessor=auto_processor,
        ):
            processor = processor_utils.get_processor(
                "local-tokenizer",
                model_name=remote_model,
            )

        self.assertIs(processor, loaded_processor)
        # 验证 AutoConfig 使用的是解析后的本地路径
        auto_config.from_pretrained.assert_called_once_with(
            local_model,
            trust_remote_code=False,
            revision=None,
        )
```

评论区精华

无 review 评论，仅 alexnails 批准（LGTM）。社区未报告争议点。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低。源码仅增加 2 行，且 URI 解析是幂等操作（已解析的路径再次调用 resolve_runai_obj_uri 会直接返回原值）。测试覆盖了关键分支。唯一需要注意：若 model_name 本是本地路径且包含特殊字符，resolve_runai_obj_uri 不应改变其值，从已有实现看该函数对非远程路径直接返回，安全。
- 影响：直接影响使用 object-storage（S3/GCS/Azure）模型路径的用户。修复前 processor 初始化（用于多模态模型）可能失败，修复后可正常加载。对其他用户无影响。受影响的模型类型包括 CLIP、DeepSeek-OCR、Qwen2-VL 等使用 get_processor 的多模态模型。
- 风险标记：暂无

关联脉络

- PR #30748 Route PD server warmup to every DP rank: 同属 object-storage / 分布式模型加载相关的基础设施改进，但功能不同。