PR #30150 完整报告

sgl-project/sglang

[diffusion][cache-dit] add dual-transformer Cache-DiT adapter specs

合并时间：2026-07-07 23:10

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30150

执行摘要

- 一句话：为双 transformer Cache-DiT 添加声明式适配器规范
- 推荐动作：该 PR 属于典型的数据驱动重构，值得阅读以理解如何通过声明式配置替代条件分支，提升代码的可扩展性和可读性。

功能与动机

避免将来添加双 transformer DiT 流水线时需要不断在 *enable_cache_on_dual_transformer* 中增加 if-else 分支，通过数据驱动方式提升可扩展性。

实现拆解

1. 定义适配器规范类：在 *cache_dit_integration.py* 中新增 DualTransformerBlockAdapterSpec 冻结数据类，包含 blocks_attr、blocks_name、forward_pattern 等字段，统一描述双 transformer 的 *cache-dit* 元数据。
2. 构建注册字典：添加全局常量 DUAL_TRANSFORMER_BLOCK_ADAPTER_SPECS，将 wan2.2 和 ideogram4 的元数据以模型名为键存储，替换原有的 _supported_dual_transformer_models 列表。
3. 改造启用函数：修改 *enable_cache_on_dual_transformer*，通过 *adapter_spec* = DUAL_TRANSFORMER_BLOCK_ADAPTER_SPECS.get(model_name) 进行查找，失败时抛异常，后续逻辑使用 *adapter_spec* 字段替代散落在函数内的硬编码值。
4. 更新单元测试桩：在 *test_cache_dit_integration.py* 中将 *cache_dit*.ForwardPattern 从 SimpleNamespace 改为自定义 _FakeForwardPattern 类，并补充 Pattern_2 常量，使类型注解 List[ForwardPattern] 能在 Python 3.10+ 正确工作。

关键文件：

- python/sglang/multimodal_gen/runtime/*cache*/*cache_dit_integration*.py（模块 DiT 集成；类别 source；类型 *core-logic*；符号 DualTransformerBlockAdapterSpec, DUAL_TRANSFORMER_BLOCK_ADAPTER_SPECS）：核心变更文件：引入 DualTransformerBlockAdapterSpec 类和注册字典，重构 *enable_cache_on_dual_transformer* 的实现。
- python/sglang/multimodal_gen/test/unit/*test_cache_dit_integration*.py（模块 测试；类别 test；类型 *test-coverage*；符号 _FakeForwardPattern）：测试配套文件：更新 ForwardPattern 桩以支持 Pattern_2 和 List[ForwardPattern] 注解。

关键符号：enable_cache_on_dual_transformer

关键源码片段

python/sglang/multimodal_gen/runtime/cache/cache_dit_integration.py

核心变更文件：引入 DualTransformerBlockAdapterSpec 类和注册字典，重构 enable_cache_on_dual_transformer 的实现。

```
@dataclass(frozen=True)
class DualTransformerBlockAdapterSpec:
    """BlockAdapter metadata for dual-transformer DiT pipelines.
    This spec covers how cache-dit should find blocks and interpret each block's forward.
    """
    blocks_attr: tuple[str, str] # Attribute names to retrieve blocks from each transformer
    blocks_name: Optional[List[str]] # Optional explicit block names per transformer
    forward_pattern: List[ForwardPattern] # Forward pattern for each (e.g., Pattern_2, Pattern_3)
    check_forward_pattern: bool # Whether to validate forward signature
    check_num_outputs: bool # Whether to check number of outputs
    has_separate_cfg: bool # Whether each transformer has its own cache config


# Registry mapping model names to their adapter specs
DUAL_TRANSFORMER_BLOCK_ADAPTER_SPECS: dict[str, DualTransformerBlockAdapterSpec] = {
    "wan2.2": DualTransformerBlockAdapterSpec(
        blocks_attr=("blocks", "blocks"),
        blocks_name=None,
        forward_pattern=[ForwardPattern.Pattern_2, ForwardPattern.Pattern_2],
        check_forward_pattern=True,
        check_num_outputs=False,
        has_separate_cfg=True,
    ),
    "ideogram4": DualTransformerBlockAdapterSpec(
        blocks_attr=("layers", "layers"),
        blocks_name=["layers", "layers"],
        forward_pattern=[ForwardPattern.Pattern_3, ForwardPattern.Pattern_3],
        check_forward_pattern=False,
        check_num_outputs=False,
        has_separate_cfg=False,
    ),
}
```

python/sglang/multimodal_gen/test/unit/test_cache_dit_integration.py

测试配套文件：更新 ForwardPattern 桩以支持 Pattern_2 和 List[ForwardPattern] 注解。

```
class _FakeForwardPattern:
    # A class (not SimpleNamespace) so it is a valid type in
    # annotations like List[ForwardPattern], matching the real Enum.
```

```
    Pattern_2 = "Pattern_2"
    Pattern_3 = "Pattern_3"


def _install_cache_dit_stub():
    # ... other stub setup ...
    cache_dit.ForwardPattern = _FakeForwardPattern # Replaces SimpleNamespace
    # Now List[ForwardPattern] works during import / type checking
```

评论区精华

gemini-code-assist[bot] 指出新数据类字段使用了 PEP 604 联合类型 “str | None”，可能在不支持该语法的 Python 版本上造成运行时错误，建议改用 Optional[List[str]]。作者 Jzz1943 采纳建议并更新了类型注解，同时指出主要动机是更精确的双transformeblocks_name类型。

- 类型注解兼容性 (correctness): 作者 Jzz1943 采纳建议，将 blocks_name 改为 Optional[List[str]]，forward_pattern 改为 List[ForwardPattern]。

风险与影响

- 风险：本次重构仅移动元数据结构，不改变运行时行为，回归风险极低。唯一的兼容性风险（Python <3.10 的联合类型语法）已在 review 中修复，且测试通过。
- 影响：对用户透明，无性能影响。对开发团队而言，新增双 transformer 模型时只需向 DUAL_TRANSFORMER_BLOCK_ADAPTER_SPECS 添加一条记录即可，无需修改函数体，降低了维护成本。
- 风险标记：低风险重构，类型兼容性修复

关联脉络

- 暂无明显关联 PR