PR #25328 完整报告

sgl-project/sglang

[diffusion] Mount Cache-DiT before torch.compile in native denoising

合并时间：2026-05-15 18:08

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25328

执行摘要

- 一句话：修复 Cache-DiT 挂载与 torch.compile 的顺序问题
- 推荐动作：建议合并。PR 定位清晰、改动简洁、性能收益显著，且已获得 reviewer 批准。值得关注的设计决策是将编译延迟到 Cache-DiT 挂载之后的方法定义，以及使用模块 id 集合避免重复编译的做法。

功能与动机

经过 profiling 发现，Cache-DiT 应在 denoising transformer 被 torch.compile 之前挂载，否则 warmup 会编译未包装的 transformer 路径，而真实请求使用 Cache-DiT 路径，导致 warmup 未能预热实际使用的路径。

实现拆解

1. 延迟 torch.compile 到 Cache-DiT 挂载后：在 DenoisingStage.__init__ 中移除 1 行私有状态变量声明，新增 _torch_compiled_module_ids 集合用于避免重复编译；在 _maybe_enable_torch_compile 方法开头增加守卫条件：若启用了 Cache-DiT 且尚未挂载，则跳过编译并返回。
2. 提取公共方法 _maybe_enable_cache_dit_and_torch_compile：将原来分散的 _maybe_enable_cache_dit 和随后对 _maybe_enable_torch_compile 的调用合并到一个新方法中，按 _maybe_enable_cache_dit 先执行、_maybe_enable_torch_compile 后执行的顺序，确保 Cache-DiT 挂载在编译之前。
3. 修改 _prepare_denoising_loop 中的调用点：将原本调用 _maybe_enable_cache_dit 后跟 _maybe_enable_torch_compile 的模式替换为调用新方法 _maybe_enable_cache_dit_and_torch_compile。
4. 调整 warmup 逻辑：当启用了 torch.compile 时，即使是 warmup 请求也允许挂载 Cache-DiT，以确保编译路径与实际运行路径一致。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py（模块 扩散模型；类别 source；类型 core-logic；符号 _maybe_enable_cache_dit_and_torch_compile）：核心变更文件，修改了 DenoisingStage 的初始化、torch.compile 守卫逻辑和 warmup 流程，新增了聚合方法 _maybe_enable_cache_dit_and_torch_compile，确保 Cache-DiT 在 torch.compile 之前挂载。

关键符号：_maybe_enable_cache_dit_and_torch_compile

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py

核心变更文件，修改了 DenoisingStage 的初始化、torch.compile 守卫逻辑和 warmup 流程，新增了聚合方法 _maybe_enable_cache_dit_and_torch_compile，确保 Cache-DiT 在 torch.compile 之前挂载。

```
# 核心改动：延迟 torch.compile，确保缓存挂载后编译
class DenoisingStage(PipelineStage, RolloutDenoisingMixin):
    def __init__(self, transformer, scheduler, pipeline=None, transformer_2=None, vae=None):
        super().__init__()
        self.transformer = transformer
        self.transformer_2 = transformer_2
        # 新增：缓存状态与已编译模块 ID 集合
        self._cache_dit_enabled = False
        self._cached_num_steps = None
        self._torch_compiled_module_ids: set[int] = set()
        # ... 原有初始化逻辑 ...
        # torch compile（此时若 Cache-DiT 未启用则跳过）
        for transformer in filter(None, [self.transformer, self.transformer_2]):
            self._maybe_enable_torch_compile(transformer)
        # ...

    def _maybe_enable_torch_compile(self, module: object) -> None:
        """
        编译模块，但在 Cache-DiT 启用且未挂载时推迟执行。
        通过 id 集合防止重复编译。
        """
        if not self.server_args.enable_torch_compile or not isinstance(module, nn.Module):
            return
        # 关键守卫：如果启用了 Cache-DiT 但尚未挂载，返回不编译
        if envs.SGLANG_CACHE_DIT_ENABLED and not self._cache_dit_enabled:
            logger.debug("Deferring torch.compile until cache-dit is enabled")
            return
        module_id = id(module)
        if module_id in self._torch_compiled_module_ids:
            return # 已编译过，跳过
        # ... 原有编译逻辑 ...
        module.compile(**compile_kwargs)
        self._torch_compiled_module_ids.add(module_id)

    def _maybe_enable_cache_dit_and_torch_compile(
        self, num_inference_steps: int | tuple[int, int], batch: Req
    ) -> None:
        """
        按序执行：先挂载 Cache-DiT，再 torch.compile。
        保证编译覆盖真实运行路径。
```

```
        """
        # 先挂载缓存（设置 self._cache_dit_enabled = True）
        self._maybe_enable_cache_dit(num_inference_steps, batch)
        # 再触发编译（此时守卫允许通过）
        for transformer in filter(None, [self.transformer, self.transformer_2]):
            self._maybe_enable_torch_compile(transformer)

    def _prepare_denoising_loop(self, batch: Req, server_args: ServerArgs):
        # ... 省略前序逻辑 ...
        if not transformer_was_loaded:
            # 加载 transformer
            loader = TransformerLoader()
            self.transformer = loader.load(...)
            # 使用聚合方法，保证顺序：先 Cache-DiT 后 torch.compile
            self._maybe_enable_cache_dit_and_torch_compile(
                cache_dit_num_inference_steps, batch
            )
        # ...
```

评论区精华

仅有一条 review 评论：mickqian 建议将 enable_cache_dit 和 _maybe_enable_torch_compile 聚合为单一方法。提交者 qimcis 采纳建议，在后续 commit 中实现了 _maybe_enable_cache_dit_and_torch_compile 方法。

- 聚合 cache_dit 与 torch.compile 方法 (design): 采纳建议，后续 commit 新增了 _maybe_enable_cache_dit_and_torch_compile 方法统一调用顺序。

风险与影响

- 风险：该 PR 修改了 warmup 和编译顺序，可能影响以下场景：
  - 非 Cache-DiT 用户：逻辑上不受影响，因为守卫条件仅在 SGLANG_CACHE_DIT_ENABLED 为 true 时触发延迟编译。
  - NPU 后端：_maybe_enable_torch_compile 中 NPU 路径不受影响，因为编译延迟后仍会执行相同逻辑。
  - 缓存一致性：_torch_compiled_module_ids 用于防止重复编译，若同一模块被不同请求使用，可能存在编译缓存未命中的风险（但 id 匹配保证了不会重复编译）。
  - 没有新增测试：变更影响了核心热路径，但未添加专门测试，回归风险中等。
  - 影响：影响范围：仅修改 denoising.py 一个文件，但涉及 Diffusion 流水线的核心 warmup 和首次请求路径。影响程度：
- 对启用 Cache-DiT 的用户：首次请求 DenoisingStage 延迟降低约 43.77%，客户端总延迟降低约 2.89%。
- 对未启用 Cache-DiT 的用户：无功能影响。
- 对团队：简化了代码逻辑，将两个需要保证顺序的操作聚合为一个方法，降低后续误用风险。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #25305 [diffusion] Fix Z-Image Cache-DiT sequence-parallel override: 同一模块 Cache-DiT 的功能修复 PR，与本 PR 同属 diffusion 下的 Cache-DiT 相关改进。