PR #24304 完整报告

sgl-project/sglang

[diffusion] feat: cache encoder results for default negative prompt

合并时间：2026-05-05 11:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24304

执行摘要

- 一句话：缓存 diffusion 默认负提示文本编码结果，减少约 54% 编码延迟
- 推荐动作：值得精读，展示了在推理引擎中引入缓存时的典型设计考量：键范围选择、引用 vs 克隆的权衡、warmup 行为。适合作为性能优化的参考案例。

功能与动机

默认负提示在 CFG 的多次请求之间通常固定不变，但当前每个请求都会重复调用完整文本编码，造成大量冗余计算。PR 旨在消除此冗余以降低请求延迟。

实现拆解

1. 添加实例级缓存字段：在 TextEncodingStage.__init__ 中引入 _negative_text_cache_key 和 _negative_text_cache_value，初始为 None。
2. 构建缓存键：提取 _build_negative_text_cache_key 方法，返回元组包含 pipeline_class_name、encoder_indices、负提示文本的冻结表示、prompt 模板的冻结表示（后续根据 review 建议移除了 max_sequence_length）。
3. 读写缓存逻辑：新增 get_or_compute_negative_text_embedding 方法，根据 is_warmup 标志决定是否查询缓存；若缓存命中则直接返回缓存的五类输出（embeds、masks、pooler 等），否则调用 encode_text 计算，并在非 warmup 时更新缓存。
4. 修改 forward 路径：将 forward 中原本直接调用 self.encode_text 编码负提示的部分替换为调用 self.get_or_compute_negative_text_embedding。
5. 添加辅助函数 stack_tensors：用于形状校验和堆叠张量。
6. 配套测试：新增 test_text_encoding_cache.py 单元测试，覆盖缓存键变化（max_sequence_length、prompt_template）和 warmup 跳过缓存。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/text_encoding.py（模块 文本编码；类别 source；类型 core-logic；符号 stack_tensors, get_or_compute_negative_text_embedding, _build_negative_text_cache_key）：核心修改文件：添加缓存实例变量、提取缓存键与方法、修改 forward 路径，实现单条目负提示缓存。

- python/sglang/multimodal_gen/test/unit/test_text_encoding_cache.py（模块 测试；类别 test；类型 test-coverage；符号 DummyTextEncodingStage, init, encode_text, make_req）：新增单元测试文件，验证缓存正确性（键变化触发重新计算、warmup 跳过缓存）。

关键符号：stack_tensors, get_or_compute_negative_text_embedding, _build_negative_text_cache_key, encode_text, make_req, test_negative_text_cache_key_tracks_encode_options, test_negative_text_cache_skips_warmup

关键源码片段

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

核心修改文件：添加缓存实例变量、提取缓存键与方法、修改 forward 路径，实现单条目负提示缓存。

```
def get_or_compute_negative_text_embedding(
    self, batch: Req, server_args: ServerArgs, all_indices: list[int]
):
    # 构建缓存键（不含 max_sequence_length，因 encode_text 不使用）
    negative_cache_key = self._build_negative_text_cache_key(
        batch, server_args, all_indices
    )
    # warmup 请求不读不写缓存，以避免预填充不相关的负提示
    use_negative_cache = not batch.is_warmup
    cached_negative = None
    if use_negative_cache:
        cached_negative = (
            self._negative_text_cache_value
            if self._negative_text_cache_key == negative_cache_key
            else None
        )
    if cached_negative is None:
        # 缓存未命中或 warmup：执行实际编码
        (
            neg_embeds_list, neg_masks_list, neg_pooler_embeds_list,
            neg_embeds_masks_list, neg_seq_lens_list,
        ) = self.encode_text(
            batch.negative_prompt, server_args,
            encoder_index=all_indices, return_attention_mask=True,
        )
        # 非 warmup 且计算完成后写入缓存
        if use_negative_cache:
            self._negative_text_cache_key = negative_cache_key
            self._negative_text_cache_value = (
                tuple(neg_embeds_list), tuple(neg_masks_list),
                tuple(neg_pooler_embeds_list), tuple(neg_embeds_masks_list),
```

```
                tuple(neg_seq_lens_list),
            )
        else:
            # 缓存命中：直接复用之前的结果
            (
                neg_embeds_list, neg_masks_list, neg_pooler_embeds_list,
                neg_embeds_masks_list, neg_seq_lens_list,
            ) = cached_negative
        return (
            neg_embeds_list, neg_masks_list, neg_pooler_embeds_list,
            neg_embeds_masks_list, neg_seq_lens_list,
        )

    def _build_negative_text_cache_key(
        self, batch: Req, server_args: ServerArgs, encoder_indices: list[int]
    ):
        # 缓存键确保覆盖影响编码结果的所有维度：
        # - 不同 pipeline 可能使用不同 tokenizer 或 encoder 集合
        # - prompt 模板和负提示文本本身内容
        return (
            server_args.pipeline_class_name,
            tuple(encoder_indices),
            self.freeze_for_dedup(batch.negative_prompt),
            self.freeze_for_dedup(batch.prompt_template),
        )
```

评论区精华

线程 1：缓存键中包含 max_sequence_length 但不被 encode_text 使用

> gemini-code-assist[bot]：batch.max_sequence_length 包含在缓存键中，但未传递给 self.encode_text，可能导致不必要的缓存未命中。 mickqian：已修复，移除了 max_sequence_length，因为该参数在此路径中不影响编码。

线程 2：缓存引用可能被后续 in-place 操作损坏

> gemini-code-assist[bot]：缓存存储的是张量引用，若下游有 in-place 修改会损坏缓存。 mickqian：已验证 LTX-2.3 下游只有读取和 cat，无 in-place 修改；克隆会额外占用 ~36Mi的存，故保留引用。通过两次请求帧级精确对比（max_abs_diff=0）确认安全性。

- 缓存键包含 max_sequence_length 但未被 encode_text 使用 (correctness): 作者移除了 max_sequence_length，确认不影响编码结果。
- 缓存存储引用可能导致后续 in-place 修改损坏 (design): 确认安全，保留引用策略。

风险与影响

- 风险：
  1. 引用安全：虽然作者验证了下游无 in-place 修改，但后续其他 pipeline 或修改可能引入此类操作，需在维护时注意。

2. 缓存键完备性：缓存键不包括 max_sequence_length 等参数，若未来编码逻辑依赖这些参数，需同步更新键。
3. 内存占用：LTX-2.3 的默认负提示嵌入 ~367 MiB，在单条目缓存下这是上限，对显存敏感的场景可能需关注。
4. warmup 行为：warmup 请求不会填充缓存，因此第一个真实请求仍会触发完整编码，但不会因 warmup 的“错误”负提示污染缓存。 - 影响：用户：使用默认负提示的 diffusion 请求（如 LTX-2.3）获得明显延迟改善（~54%），自定义负提示请求不受影响。对于多次相同负提示的连续请求收益最大。系统：增加约 367 MiB 显存占用（LTX-2.3 场景），在单条目缓存下可控。warmup 流程不变。团队：提供可复用的缓存模式，未来其他 pipeline 或正提示也可以借鉴。

- 风险标记：缓存引用安全，缓存键完备性，内存占用 367MiB

关联脉络

- 暂无明显关联 PR