PR #27023 完整报告

sgl-project/sglang

[diffusion] Optimize LingBot realtime transformer path

合并时间：2026-06-02 18:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27023

执行摘要

- 一句话：优化 LingBot 实时 transformer 路径，缓存 RoPE 和时间嵌入
- 推荐动作：值得精读，特别是缓存设计和 update_cache_only 参数的使用方式。建议作者补充性能对比数据（benchmark），以量化优化效果。

功能与动机

LingBot 实时推理需要在每次前向传播中计算 RoPE 和时间嵌入，这些计算在 sequence-sharded 模式下可被缓存以减少开销。同时 CI 案例解析器不支持 ONE_GPU_CASES.append(CaseFactory()) 这种常见模式，导致新增案例被忽略。

实现拆解

1. Transformer 路径优化（lingbot_world.py）：新增 _prepare_cached_rope_for_sequence_shard 和 _prepare_cached_time_embeddings 方法，利用 RealtimeCausalDiTState.runtime_cache 缓存 RoPE 元组和时间嵌入；在 CausalLingBotSelfAttention.forward 和 LingBotTransformerBlock.forward 中新增 update_cache_only 参数，当该参数为 true 时，只执行 KV 缓存更新并提前返回。
2. 缓存状态管理（causal_state.py）：在 RealtimeCausalDiTState 中新增 runtime_cache: dict，用于存放实时缓存对象；dispose 方法中调用 clear()。
3. 上下文时间步修正（lingbot_world_causal_denoising.py）：将 _update_causal_context_cache 中的 current_timestep 从 0 改为 -1，以符合实际语义。
4. CI 案例解析增强（diffusion_case_parser.py）：扩展 DiffusionTestCaseVisitor 以解析模块级 append 调用，新增 _process_expr 和 _extract_factory_case_id 方法，支持识别工厂函数返回的 case_id。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/lingbot_world.py（模块 模型层；类别 source；类型 core-logic；符号 _prepare_cached_rope_for_sequence_shard, _prepare_cached_time_embeddings, CausalLingBotSelfAttention.forward, LingBotTransformerBlock.forward）：核心优化：新增缓存方法，修改 attention 和 transformer block forward 支持 update_cache_only。
- scripts/ci/utils/diffusion/diffusion_case_parser.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 visit_Module, _process_expr, _extract_factory_case_id）：CI 增

强：支持解析 append 调用和工厂函数，确保测试案例被正确识别。

- python/sglang/multimodal_gen/runtime/realtime/causal_state.py（模块 实时状态；类别 source；类型 data-contract；符号 init, dispose）：数据结构扩展：新增 runtime_cache 字段用于存放缓存对象。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/lingbot_world/lingbot_world_causal_denoising.py（模块 扩散管道；类别 source；类型 data-contract）：微调：更正 current_timestep 值从 0 改为 -1。

关键符号：_prepare_cached_rope_for_sequence_shard, _prepare_cached_time_embeddings, CausalLingBotSelfAttention.forward, LingBotTransformerBlock.forward, RealtimeCausalDiTState.init, RealtimeCausalDiTState.dispose, DiffusionTestCaseVisitor.visit_Module, DiffusionTestCaseVisitor._process_expr, DiffusionTestCaseVisitor._extract_factory_case_id

关键源码片段

python/sglang/multimodal_gen/runtime/models/dits/lingbot_world.py

核心优化：新增缓存方法，修改 attention 和 transformer block forward 支持 update_cache_only。

```
def _prepare_cached_rope_for_sequence_shard(
    self, *, forward_batch, local_seq_len, token_start, frame_stride,
    post_patch_width, device
) -> tuple[torch.Tensor, ...]:
    # 通过 request cache 缓存 RoPE，key 包含所有相关参数
    cache = self._get_request_cache(forward_batch, "lingbot_sequence_shard_rope")
    cache_key = (local_seq_len, token_start, frame_stride, post_patch_width, device.type, device.
    index)
    if cache is not None and cache_key in cache:
        return cache[cache_key]

    freqs_cos, freqs_sin = self._compute_rope_for_sequence_shard_with_offset(
        local_seq_len, token_start, frame_stride, post_patch_width, device
    )
    freqs_cos = freqs_cos.float()
    freqs_sin = freqs_sin.float()
    freqs_cis = (freqs_cos, freqs_sin)
    if _is_cuda:
        # 对于 CUDA 额外缓存融合后的 cos+sin，减少后续拼接开销
        freqs_cis = (
            freqs_cos,
            freqs_sin,
            torch.cat([freqs_cos.contiguous(), freqs_sin.contiguous()], dim=-1),
        )
    if cache is not None:
        cache.clear() # 每次更新只保留一个 key 避免内存膨胀
```

```
        cache[cache_key] = freqs_cis
    return freqs_cis
```

scripts/ci/utils/diffusion/diffusion_case_parser.py

CI 增强：支持解析 append 调用和工厂函数，确保测试案例被正确识别。

```
def _process_expr(self, node: ast.AST):
    # 解析类似 ONE_GPU_CASES.append(SomeFactory()) 的调用
    if not isinstance(node, ast.Call):
        return
    if not isinstance(node.func, ast.Attribute) or node.func.attr != "append":
        return
    list_name = node.func.value.id
    if list_name not in CASE_LIST_TO_SUITE:
        return
    if len(node.args) != 1:
        return
    case_id = self._extract_case_id_from_call(node.args[0])
    if case_id:
        self.cases.setdefault(list_name, []).append(case_id)

def _extract_factory_case_id(self, node: ast.FunctionDef) -> Optional[str]:
    # 从工厂函数的 return 中提取 case_id
    for child in ast.walk(node):
        if not isinstance(child, ast.Return) or child.value is None:
            continue
        case_id = self._extract_case_id_from_call(child.value)
        if case_id:
            return case_id
    return None
```

python/sglang/multimodal_gen/runtime/realtime/causal_state.py

数据结构扩展：新增 runtime_cache 字段用于存放缓存对象。

```
class RealtimeCausalDiTState(BaseRealtimeState):
    """persist causal DiT cache and frame position across realtime chunks"""

    def __init__(self):
        super().__init__()
        self.kv_cache = None
        self.crossattn_cache = None
        self.runtime_cache: dict = {} # 新增：按名称存储临时缓存（如 RoPE）
        self.current_chunk_start_frame: int = 0
        self.chunk_idx: int = 0

    def dispose(self) -> None:
        self.kv_cache = None
        self.crossattn_cache = None
        self.runtime_cache.clear() # 确保缓存不会跨请求泄漏
```

```
self.current_chunk_start_frame = 0
self.chunk_idx = 0
```

评论区精华

PR 无实质性 review 讨论，仅包含自动化标签触发和 quota 达到提示。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 内存占用：新增的 runtime_cache 会缓存 RoPE 和时间嵌入，在长序列或多请求场景下可能增加内存压力（但通过 key 哈希和 clear() 控制）。
  2. 行为兼容性：新增 update_cache_only 参数需要调用方传递，若忘记传递则默认 False，行为不变；但若在已使用该参数的地方未更新，可能导致意外行为。
  3. CI 解析器：对 append 文档的解析依赖于 AST 模式匹配，若工厂函数定义复杂（如返回多个 case_id）可能导致漏解析。 - 影响：对用户：LingBot 实时推理性能提升（减少重复计算），TTFT 应有所下降。对系统：新增的 cache 机制增加了内存使用，但可控。对团队：CI 中 append 方式的案例能被正确执行，避免测试遗漏。影响范围限定在 diffusion 模块和 CI 脚本。 - 风险标记：核心路径变更，新增缓存机制，CI 解析逻辑变更

关联脉络

- PR #26954 [diffusion] misc: 引入了 LingBot World 实时扩散管道与 Transformer，本 PR 是对其的后续性能优化。
- PR #26959 [diffusion] add WebUI: 为 LingBot 实时 WebUI 提供后端支持，本 PR 优化了后端关键路径。