PR #29271 完整报告

sgl-project/sglang

fix: make write_token dynamic

合并时间：2026-07-01 13:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29271

执行摘要

- 一句话：修复 MLX 后端 write_token 越界写入 bug
- 推荐动作：值得精读，这是一个典型的边界修复，代码改动简洁且与现有数据结构的设计一致，测试覆盖完整。可作为 MLX 后端类似改动的参考。

功能与动机

MLX 后端的 ContiguousAttentionKVCache 为每个请求分配固定大小的 KV 缓冲区（默认 max_seq_len=4096）。prefill 路径的 update_and_fetch 已有动态扩容（_grow），但 decode 路径的 write_token 未检查容量，导致长序列生成时发生越界写，损坏 KV cache。PR body 明确指出该问题并引用了一个已存在的 TODO。

实现拆解

1. 修改核心 KV Cache 类：在 python/sglang/srt/hardware_backend/mlx/kv_cache/attention_kv_cache.py 的 write_token 方法中，在写入前增加容量检查：计算 end = self.offset + 1，如果 end > self.max_seq_len 则调用已有的 _grow(end) 方法进行扩容（每次翻倍），然后执行切片赋值并更新 self.offset。该方式与 update_and_fetch 的扩容逻辑完全一致，保证了两个写入路径的行为统一。
2. 清理遗留 TODO：在 python/sglang/srt/hardware_backend/mlx/model_runner.py 的 decode_batch_start_chained 方法中，移除了此前标记需要修复 write_token 动态扩容的 TODO 注释（3 行）。
3. 添加单元测试：在 test/registered/unit/hardware_backend/mlx/test_attention_patching.py 中新增 test_write_token_grows_buffer_past_max_seq_len 方法，创建一个初始 max_seq_len=4 的 KV cache，通过 write_token 写入 2*max_seq_len+1 个 token，验证 offset、max_seq_len、keys/values 的 shape 以及每个 token 内容的正确性。

关键文件：

- python/sglang/srt/hardware_backend/mlx/kv_cache/attention_kv_cache.py（模块 KV Cache；类别 source；类型 core-logic）：核心修复文件：ContiguousAttentionKVCache.write_token 增加动态扩容逻辑，与 update_and_fetch 保持一致。
- test/registered/unit/hardware_backend/mlx/test_attention_patching.py（模块 测试；类别 test；类型 test-coverage；符号 test_write_token_grows_buffer_past_max_seq_len）

：新增测试方法，覆盖 write_token 动态扩容场景，验证数据完整性。

- python/sglang/srt/hardware_backend/mlx/model_runner.py（模块 模型运行器；类别 source；类型 data-contract）：移除了已解决 bug 的 TODO 注释，保持代码整洁。

关键符号：ContiguousAttentionKVCache.write_token, test_write_token_grows_buffer_past_max_seq_len

关键源码片段

python/sglang/srt/hardware_backend/mlx/kv_cache/attention_kv_cache.py

核心修复文件：ContiguousAttentionKVCache.write_token 增加动态扩容逻辑，与 update_and_fetch 保持一致。

```
def write_token(self, k: mx.array, v: mx.array) -> None:
    """Write one token. k, v shape: (1, n_kv_heads, 1, head_dim)."""
    # 计算写入后的结束位置
    end = self.offset + 1
    # 如果超出当前容量，则调用 _grow 扩容（翻倍策略，与 update_and_fetch 一致）
    if end > self.max_seq_len:
        self._grow(end)
    # 切片赋值写入 KV
    self.keys[:, :, self.offset : end, :] = k
    self.values[:, :, self.offset : end, :] = v
    # 更新偏移量
    self.offset = end
```

test/registered/unit/hardware_backend/mlx/test_attention_patching.py

新增测试方法，覆盖 write_token 动态扩容场景，验证数据完整性。

```
def test_write_token_grows_buffer_past_max_seq_len(self):
    max_seq_len = 4
    cache = ContiguousAttentionKVCache(
        n_kv_heads=1, head_dim=2, max_seq_len=max_seq_len, dtype=mx.float32
    )
    n_tokens = max_seq_len * 2 + 1 # 确保至少触发一次扩容

    for t in range(n_tokens):
        k = mx.full((1, 1, 1, 2), t, dtype=mx.float32)
        v = mx.full((1, 1, 1, 2), -t, dtype=mx.float32)
        cache.write_token(k, v)

    # 验证 offset 和 max_seq_len 已更新
    self.assertEqual(cache.offset, n_tokens)
    self.assertGreaterEqual(cache.max_seq_len, n_tokens)

    keys, values = cache.get_kv()
    mx.eval(keys, values)
    # 验证 KV shape 正确
    self.assertEqual(keys.shape, (1, 1, n_tokens, 2))
```

```
        self.assertEqual(values.shape, (1, 1, n_tokens, 2))
        # 验证每个 token 的内容，包括扩容前写入的 token 是否被正确保留
        for t in range(n_tokens):
            self.assertEqual(keys[0, 0, t, 0].item(), float(t))
            self.assertEqual(values[0, 0, t, 0].item(), float(-t))
```

评论区精华

Review 中获得两位 reviewer 的 APPROVAL。jlee5814 评论 "LGTM. write_token now matches update_and_fetch."，确认了实现一致性。yeahdongcn 提示需要 rebase 以解决 CI 单元测试失败（关联 PR #29311）。

- 暂无高价值评论线程

风险与影响

- 风险：本 PR 仅影响 MLX 后端的 ContiguousAttentionKVCache，且修改行为与已有的 update_and_fetch 一致，风险较低。_grow 方法已被复用，其内部的 valid-prefix copy 逻辑在 MLX 延迟计算图下正常工作。test_write_token_grows_buffer_past_max_seq_len 提供了边界覆盖。需要注意：若链式 decode 路径在扩容后重新引用缓存对象，由于扩容原地替换了 keys/values 属性，链式路径能透明感知扩容后的 buffer。
- 影响：仅在 MLX 后端生效，修复了长序列生成时 KV cache 可能损坏的 bug，使 decode 路径支持任意长度的生成序列。对现有模型无性能或精度影响，因为扩容仅在超出初始 max_seq_len 时触发，且使用与之前相同的翻倍策略。
- 风险标记：MLX 专用修改，有测试覆盖

关联脉络

- PR #29311 Address unit test failure related to MLX: yeahdongcn 在 issue 评论中提及本 PR 的 CI 测试失败与 #29311 相关，建议 rebase。