PR #22645 完整报告

sgl-project/sglang

env: add knob to control SWA eviction interval

合并时间：2026-04-14 06:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22645

执行摘要

本 PR 新增环境变量 SGLANG_SWA_EVICTION_INTERVAL_MULTIPLIER，允许用户调节滑动窗口注意力 (SWA) 的淘汰频率，以平衡内存浪费与淘汰计算开销。核心修改涉及调度批处理逻辑，将固定淘汰间隔改为可配置的倍数。虽然 review 中指出了边界情况风险，但代码未完全处理，建议用户使用时避免设置过小乘数。

功能与动机

当前 SWA 淘汰机制每 sliding_window_size 个解码步骤执行一次，导致每个请求在淘汰间隔内最多持有 2 * sliding_window_size 个 SWA 令牌，是滑动窗口实际需要的两倍，造成令牌浪费。PR body 明确表示："Adding a knob to trade-off between the token waste and per-forward overhead." 即新增调节旋钮，让用户能在内存占用与淘汰开销之间进行权衡。

实现拆解

实现分为两个模块：

1. 环境变量定义（python/sglang/srt/environ.py）： python SGLANG_SWA_EVICTION_INTERVAL_MULTIPLIER = EnvFloat(1.0) 新增浮点型环境变量，默认值 1.0 保持向后兼容。
2. 淘汰逻辑更新（python/sglang/srt/managers/schedule_batch.py）： python page_size = self.tree_cache.page_size eviction_interval = max(page_size, int(sliding_window_size * envs.SGLANG_SWA_EVICTION_INTERVAL_MULTIPLIER.get()),) eviction_interval = (eviction_interval // page_size) * page_size if req.decode_batch_idx % eviction_interval == 1: self._evict_swa(req, req.seqlen - 1) 淘汰间隔取页面大小和 sliding_window_size * 乘数 的最大值，并向下对齐到页面大小的整数倍，然后替换原固定间隔条件。

评论区精华

review 中唯一的技术讨论来自 gemini-code-assist[bot]，聚焦于淘汰条件的边界情况：

> "The condition req.decode_batch_idx % eviction_interval == 1 will never be true if eviction_interval is 1. This can happen if page_size is 1 and the multiplier is set to a very small value. In this case, SWA eviction would never trigger during decoding, which contradicts the user's intent of increasing eviction frequency."

建议修改为 `if req.decode_batch_idx > 0 and req.decode_batch_idx % eviction_interval == 1:` 以保留跳过 `decode_batch_idx == 0` 的逻辑并处理间隔为 1 的情况。但最终代码未采纳此建议，仅更新了注释。hnyls2002 直接批准，未进一步讨论。

风险与影响

- 逻辑风险：当 *eviction_interval* 计算为 1 时（例如 *page_size*=1 且乘数很小），淘汰条件永不成立，SWA 淘汰永不执行，可能导致内存累积或性能下降。
- 兼容性：默认乘数 1.0 保持原行为，但用户若设置极小值（如 0.1）可能意外触发上述问题。
- 测试覆盖：PR 未包含单元测试或性能基准，无法验证不同乘数下的正确性和权衡效果。
- 影响范围：对现有部署无影响，高级用户可通过环境变量优化内存与计算开销，但需谨慎设置参数。

关联脉络

从近期历史 PR 看，本 PR 与 PR#22730（环境变量重构）在代码模式上相关，都属于通过环境变量增强系统可配置性。SWA 淘汰机制是滑动窗口注意力优化的关键部分，此次变更反映了团队在内存管理和计算开销之间寻求更细粒度平衡的趋势。结合 PR#22122（LoRA MoE 虚拟专家）等性能优化 PR，可见仓库持续关注推理效率的微调。