PR #34189 完整报告

sgl-project/sglang

[DSV4] Fix silent KV corruption when speculative draft tokens > 4

合并时间：2026-08-10 06:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34189

执行摘要

- 一句话：修复 DSV4 投机 draft>4 时压缩环静默写坏 KV
- 推荐动作：该 PR 值得精读，尤其适合负责 DeepSeek-V4 压缩注意力、投机解码和 CUDA 内核开发的工程师。值得关注的设计决策：将回滚保护 pad 从批次的属性改为环的属性（单一事实来源）、容量校验放在启动时而非规划器中（明确职责边界）、用“逐步骤写满自身扩展区间”归纳证明驻留充分性（避免多步回放测试的复杂度）、以及“回归测试必须能在修复前失败”的验证纪律。

功能与动机

PR body 明确描述：投机 verify 批次按乐观 seq_len = prefix_len + num_draft_tokens 规划写入，但实际回滚到 prefix_len + accept_len，因此每个已提交 token 必须保持驻留。c_plan.cuh 中 kMaxMTPDraftTokens = 4 的硬编码上限导致 --speculative-num-draft-tokens > 4 时计划欠写环形缓冲，后续压缩读取陈旧槽位——无声失败、无非法访问、无 NaN、无断言，只是压缩状态错误。同时 CPU 主机规划路径完全缺失 pad 项，与 GPU 路径不一致。

实现拆解

变更分四步展开：

1. 内核规划器 pad 推导（python/sglang/kernels/jit/csrc/deepseek_v4/c_plan.cuh）：将 mtp_pad 从硬编码 std::min(ring_size - compress_ratio, kMaxMTPDraftTokens) 改为 ring_size > window_size ? ring_size - window_size + 2 : 0。推导依据：写位置 w 会别名到 w - ring_size，未来压缩最早需要的位置是 prefix_len - window_size + 2（下一批次至少提交 1 个 token，且 run_prefill 在写内核之前启动压缩内核，批次自身压缩读的是写前环）。. 非投机环恰好一个窗口宽，此时 pad 为 0，写入集不变。同时主机循环路径的 first_w_pos增加seq_len - mtp_pad` 项，使 CPU 与 GPU 两条规划路径一致。
2. 容量辅助函数（python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py）：新增 get_compress_state_write_pad(compress_ratio, ring_size)，与 c_plan.cuh 公式镜像，供 Python 侧校验使用；ring_size <= window_size 时钳制为 0，覆盖 online c128 环大小为 1 的场景。
3. 启动时容量校验（python/sglang/srt/model_executor/pool_configurator.py）：在 DSV4PoolConfigurator.__init__ 中当 is_speculative 时调用新增

_assert_ring_serves_draft_tokens(max_speculative_num_draft_tokens)。之所以放启动时而非规划器内，是因为规划器无法区分超配置的 verify 批次与普通长 prefill；环只定尺寸一次，必须服务最大的自适应档位。当前 c4 环支持 D <= 10，c128 环支持 D <= 130，超过即启动失败并给出明确报错信息。

4. 测试配套：新增核级测试 test/registered/kernels/ops/attention/test_deepseek_v4_compress_plan_draft_pad.py（约 8 秒 CI）和 CPU 单测 test/registered/unit/mem_cache/test_dsv4_compress_write_pad.py（约 1 秒）；修改 python/sglang/test/kits/attention_unittest/attention_methods/dsv4_attention.py，使测试 fixture 在设置 speculative_num_draft_tokens 时正确声明 EAGLE 算法。

关键文件：

- python/sglang/kernels/jit/csrc/deepseek_v4/c_plan.cuh（模块 JIT 内核；类别 source；类型 core-logic；符号 plan_compress_prefill, PrefillParams）：修复核心所在：mtp_pad 从硬编码 4 改为由环容量推导，并统一 CPU/GPU 两条规划路径的 first_w_pos 计算，消除静默 KV 损坏的根源。
- python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 get_compress_state_write_pad）：新增 get_compress_state_write_pad 辅助函数，作为 Python 侧与 c_plan.cuh 镜像的单一事实来源，供启动校验使用；ring_size <= window_size 时钳制为 0，覆盖 online c128 环大小为 1 的场景。
- python/sglang/srt/model_executor/pool_configurator.py（模块 池配置器；类别 source；类型 data-contract；符号 _assert_ring_serves_draft_tokens）：新增 _assert_ring_serves_draft_tokens 启动校验，在环只定尺寸一次的前提下用最大自适应档位校验容量，超配置服务器启动即失败，而非运行时静默损坏。
- test/registered/kernels/ops/attention/test_deepseek_v4_compress_plan_draft_pad.py（模块 压缩写计划；类别 test；类型 test-coverage；符号 _window_size, _max_draft_tokens, _written_positions, TestCompressWritePlanDraftPad）：核级回归测试：对环可服务的每个 draft 数 × seq_len % compress_ratio 的余数组合断言 plan_w 覆盖全部 [prefix_len, seq_len)，并钉死 CPU/GPU 两条规划路径一致；修复前运行有 19 个失败用例，证明测试有效。
- test/registered/unit/mem_cache/test_dsv4_compress_write_pad.py（模块 压缩写 pad；类别 test；类型 test-coverage；符号 TestCompressStateWritePad, test_pad_is_zero_without_speculation, test_pad_matches_speculative_ring_capacity, test_pad_is_zero_for_rings_below_one_window）：CPU 单测：钉死 get_compress_state_write_pad 的契约（非投机为 0、投机环 10/130、小于一个窗口的环钳制为 0），防止 Python 侧与内核侧公式漂移。
- python/sglang/test/kits/attention_unittest/attention_methods/dsv4_attention.py（模块 注意力测试；类别 test；类型 test-coverage）：测试 fixture 修正：设置 speculative_num_draft_tokens 时声明 EAGLE 算法，swa_page_size 改为与 page_size 一致，保证注意力单测在投机配置下走正确路径。

关键符号：plan_compress_prefill, get_compress_state_write_pad, _assert_ring_serves_draft_tokens

关键源码片段

python/sglang/kernels/jit/csrc/deepseek_v4/c_plan.cuh

修复核心所在：mtp_pad 从硬编码 4 改为由环容量推导，并统一 CPU/GPU 两条规划路径的 first_w_pos 计算，消除静默 KV 损坏的根源。

```
// 该片段来自 plan_compress_prefill() 内 mtp_pad 的推导与主机循环路径的写入起点计算。
// 修复前此处是硬编码：constexpr int32_t kMaxMTPDraftTokens = 4; 导致 draft > 4 时欠写环。

// Write pad: trailing tokens kept resident so a verify batch's committed tail survives
// any accept length. Zero without speculation -- nothing rolls back, and the ring is
// then exactly one window wide. Otherwise the ring bounds it: a write at `w` aliases
// onto `w - ring_size`, and the earliest position a future compression still needs is
// `prefix_len - window_size + 2` (the next batch commits >= 1 token, and `run_prefill`
// launches the compress kernel before the write kernel, so a batch's own compressions
// read the pre-write ring). Padding past the extend range is harmless: the loops only
// span `[prefix_len, seq_len)`.
// 中文注释补一句：pad 是“环的属性”而非“批次的属性”，因此三条规划路径共享同一推导。
const auto mtp_pad = ring_size > window_size ? ring_size - window_size + 2 : 0;

// 主机循环路径（CPU plan）：修复前完全缺失 pad 项，与 GPU kernel0 路径产生分歧。
const int32_t last_c_pos = seq_len / compress_ratio * compress_ratio;
// 写起点取“上一压缩窗”与“扣除 pad 的乐观尾端”两者中更靠前者，保证已提交 token 驻留。
const int32_t first_w_pos =
    std::min(last_c_pos - (is_overlap ? compress_ratio : 0), seq_len - mtp_pad);
```

python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py

新增 get_compress_state_write_pad 辅助函数，作为 Python 侧与 c_plan.cuh 镜像的单一事实来源，供启动校验使用；ring_size <= window_size 时钳制为 0，覆盖 online c128 环大小为 1 的场景。

```
def get_compress_state_write_pad(compress_ratio: int, ring_size: int) -> int:
    """Largest draft-token count this ring can serve; mirrors `mtp_pad` in `c_plan.cuh`
    (the bound is derived there). Zero for a non-speculative ring, which is exactly one
    window wide."""
    # c4 压缩一次读两个 chunk（overlap），窗口为 2 * compress_ratio；c128 不 overlap。
    window_size = compress_ratio * (2 if compress_ratio == 4 else 1)
    # ring_size <= window 时环不足一个窗口（如 online c128 环大小为 1），钳制为 0
    # 而非负数，避免启动校验误判；返回值与内核侧 mtp_pad 公式保持一致。
    return ring_size - window_size + 2 if ring_size > window_size else 0
```

python/sglang/srt/model_executor/pool_configurator.py

新增 _assert_ring_serves_draft_tokens 启动校验，在环只定尺寸一次的前提下用最大自适应档位校验容量，超配置服务器启动即失败，而非运行时静默损坏。

```
def _assert_ring_serves_draft_tokens(self, num_draft_tokens: int) -> None:
    """A verify batch writes its whole optimistic tail into the ring, so ring
    capacity bounds the draft count."""
    # 分别校验 c4 与 c128 两类压缩层；环在启动时定尺寸一次，
```

```
        # 因此必须服务最大的自适应投机档位（max_speculative_num_draft_tokens）。
        for compress_ratio, ring_size, num_layers in (
            (4, self.c4_ring_size, self.num_layers_ca4),
            (128, self.c128_ring_size, self.num_layers_ca128),
        ):
            if num_layers == 0:
                continue
            if compress_ratio == 128 and envs.SGLANG_OPT_USE_ONLINE_COMPRESS.get():
                # online c128 按 draft 存状态而非环，单独定尺寸，跳过容量校验。
                continue
            max_draft_tokens = get_compress_state_write_pad(compress_ratio, ring_size)
            assert num_draft_tokens <= max_draft_tokens, (
                f"speculative_num_draft_tokens={num_draft_tokens} exceeds what the c{compress_
                ratio} "
                f"compress state ring can keep resident (ring_size={ring_size} serves at most "
                f"{max_draft_tokens} draft tokens). Lower the draft count, or grow the ring in "
                f"get_compress_state_ring_size()."
            )
```

评论区精华

本 PR 无 review 评论，关键讨论均沉淀在 PR body 与测试文档字符串中：

- 关于“回归测试必须能在修复前失败”的验证方法：作者直接对修复前规划器运行新测试，得到 19 个失败用例，确认测试有效性。
- 关于风险暴露面的分析：c4 从 D=5 开始丢失位置（恰在 D > kMaxMTPDraftTokens 边界，边界尖锐）；c128 仅在接近环容量时破坏（first_w_pos 通常取 last_c_pos 已足够覆盖）。
- 关于端到端验证的局限：GSM8K/GPQA 精度数据无回归，但作者明确指出“这不是修复生效的证据”，并发批次的非确定性使噪声高于效应量；opaque-id 回显 200/200 精确匹配。
- 关于容量边界为何放到启动时：规划器无法区分超配置 verify 批次与普通长 prefill，因此超配置服务器应在启动时失败而非运行时损坏状态。
- 关于 c4 环扩容的权衡：提高 c4 上限需要增大环，会放大 c4 状态池，此权衡留给后续 PR。
- 回归测试有效性：修复前必须失败 (testing): 测试文件对修复前代码真实失败，验证了测试的有效性；c4 从 D=5 开始丢失位置的边界与 kMaxMTPDraftTokens=4 精确对齐，证明修复边界尖锐。
- 端到端精度数据能否证明修复生效 (question): 端到端不可见修复是预期的：被跳过的位置是 draft 尾的头部，连续 verify 批次写入集高度重叠，下一步通常重写上一步跳过的位置；损坏需要位置持续未写直到压缩真正读取。
- 容量边界为何放在启动时而非规划器 (design): 采用 DSV4PoolConfigurator._assert_ring_serves_draft_tokens 启动断言，报错信息给出明确的降级 / 扩容指引。
- 提高 c4 环容量的权衡 (design): 扩容权衡留出本 PR 范围，修复优先保证正确性，内存成本权衡后续单独处理。

风险与影响

- 风险：主要风险点如下：
- 核心路径变更：c_plan.cuh 是 DSV4 压缩注意力写计划内核，所有 prefill/verify 批次都经过它。虽然本 PR 用测试覆盖了 C4/C128 两种压缩比和 CPU/GPU 两条路径，但非投机路径“写入集不变”的断言依赖 window_size 与 ring_size 的精确关系，未来若改动 get_compress_state_ring_size 或压缩窗口定义，可能引入回归。
- 启动时断言可能过严：_assert_ring_serves_draft_tokens 使用 max_speculative_num_draft_tokens（最大自适应档位），若用户配置了自适应投机且最大档位超过环容量但实际很少使用，服务器将直接拒绝启动；这是有意设计，但属于行为变更，需要文档 / 报错信息足够清晰。
- online c128 兼容性：_assert_ring_serves_draft_tokens 对 SGLANG_OPT_USE_ONLINE_COMPRESS 下的 c128 跳过校验（环塌缩为 1，pad 钳制为 0），但 online 路径本就不支持普通 MTP，需确认跳过不会掩盖配置错误。
- 性能影响：作者论证无额外内存流量、无新内核启动，投机 verify 批次写入更多行是修复本身；但写入行数增加在环接近满时可能增加写放大，需在真实负载下观察。
- 测试覆盖：test_cpu_and_gpu_planner_agree 只覆盖 prefix_len 为 512/513/515 三种情况，且 draft 数只取 1、4、max_d；plan_w 解码依赖 ragged_id 幸存这一实现细节，若内核布局变化测试会失真。
- 影响：影响范围集中在 DeepSeek-V4 架构 + 投机解码（MTP/EAGLE）场景：
- 用户影响：使用 --speculative-num-draft-tokens > 4 的 DSV4 服务器此前会静默产生错误压缩状态（错误 KV），本 PR 修复后输出正确；超过环容量的配置（c4 超过 10、c128 超过 130）会在启动时直接失败并给出清晰指引，避免带病运行。
- 系统影响：压缩写计划 pad 由编译期常量变为运行时推导，所有规划路径行为统一；非投机场景写入集不变，无行为回归。
- 团队影响：为后续调整投机 draft 数提供了明确的容量契约（启动校验 + 单元测试），降低配置错误排查成本；测试方法论（“回归测试必须能在修复前失败”）对团队有示范价值。
- 风险标记：核心路径变更，静默数据损坏修复，新增启动时配置校验，测试覆盖依赖内核布局细节，端到端验证噪声高于效应量

关联脉络

- PR #33872 （被替代的原始 PR，未在当前列表中）：PR body 首行声明 'Supersedes #33872'，本 PR 是前者的重做版本。
- PR #34186 [CI] Key scheduled CUDA suites by runner_config instead of hand-written jobs: CI 基础设施正在向注册式测试调度迁移，本 PR 新增的核级测试使用 register_cuda_ci 注册，与 34186 的调度体系一致。
- PR #34147 [AMD] [CI] Register the DeepSeek-V4-Pro-DSpark MI35x nightly job so its suite actually runs: 同为 DSV4 系列 CI 修复，说明 DSV4 测试注册与调度是该阶段持续关注点。
- PR #34043 [srt] Fix sconv state memory corruption on specdec: 同为投机解码下状态内存损坏类 bugfix，主题相关：投机路径的状态生命周期与复用正确性。

- PR #34133 config: derive the runner's DCP topology from its ParallelState: 同为 DSV4 相关配置派生重构，反映了将派生值集中到单一事实来源（Single Source of Truth）的演进方向。