PR #34284 完整报告

sgl-project/sglang

fix(scheduler): track max prefill batch size over recent real admissions

合并时间：2026-08-15 10:48

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34284

执行摘要

- 一句话：滑动窗口跟踪 prefill 批大小高水位，修复 TPOT 劣化
- 推荐动作：该 PR 值得精读，尤其是 RecentPrefillBatchSizeTracker 和 _estimate_attempted_prefill_bs 的设计：用滑动窗口替代指数衰减来老化高水位，并在无法获得真实批大小时用保守上界估计，是一种可复用的调度状态维护模式。建议关注 max_prefill_bs 类型变化对下游判断的影响，以及窗口大小在不同负载下的调参经验。

功能与动机

PR body 明确指出：Scheduler.max_prefill_bs 原先每个 scheduler pass 衰减一次，而值在传给 PrefillDelayer 前被转成整数，因此像 2 这样小但有效的高水位在大量 decode-only 或 idle pass 之间会迅速变成 0。一旦 max_prefill_bs 变为 0，all 路径的 slot_condition 就失去效果，prefill 工作会干扰 decode 批次，导致 TPOT 下降。作者在 issue 评论中也说明这是 #32880 的 follow-up，目标是保留让过期的大峰值失效的能力，同时把老化过程与 scheduler-pass 频率解耦。

实现拆解

1. 新增 RecentPrefillBatchSizeTracker（python/sglang/srt/managers/prefill_delayer.py）：用 collections.deque(maxlen=window_size) 保存最近 N 次非空 prefill 尝试的批大小；max_prefill_bs 属性返回窗口内最大值（默认 0）；observe_attempt 校验输入必须为正整数，否则抛出 ValueError。
2. 改造 PrefillDelayerSinglePassExecutor（同文件）：finalize 的入参从 actual_prefill: bool 改为 actual_prefill_bs: int，返回实际批大小；若被拒绝（传入 0），则返回 _attempted_prefill_bs 估计值。新增 _estimate_attempted_prefill_bs 方法：在非 DP attention 场景先把 max_running_requests 按 dp_size 折算为本地槽位数，然后取 min(waiting_queue_len, free_slots) 作为保守估计。negotiate_should_allow_prefill 在 local_prefillable=True 时先更新该估计值的上界。
3. 接入 Scheduler（python/sglang/srt/managers/scheduler.py）：init_schedule_policy 中创建 RecentPrefillBatchSizeTracker，窗口大小取 envs.SGLANG_PREFILL_DELAYER_MAX_PREFILL_BS_WINDOW_SIZE；max_prefill_bs 类型从 float 改为 int。get_new_batch_prefill 删除每 pass 的 self.max_prefill_bs *= 0.998 衰减，改为在 finalize 返回正数时调用 tracker.observe_attempt 更新高水位；_get_new_batch_prefill_raw 中删除旧的 self.max_prefill_bs = max(self.max_prefill_bs,

len(can_run_list)) 行。

4. 新增环境变量（python/sglang/srt/environ.py）：SGLANG_PREFILL_DELAYER_MAX_PREFILL_BS_WINDOW_SIZE，默认 16。
5. 测试配套：新增 test/registered/unit/managers/test_prefill_delayer_high_watermark.py，覆盖峰值过期、重复小峰值保持有效、被拒绝非空尝试推进窗口、估计值打印与空值拒绝；三个 NPU 性能测试（Kimi-K2.6、MiniMax-M2.5、Qwen3-6.27B）通过环境变量将窗口调大到 64。

关键文件：

- python/sglang/srt/managers/prefill_delayer.py（模块 预填延迟器；类别 source；类型 dependency-wiring；符号 RecentPrefillBatchSizeTracker, init, max_prefill_bs, observe_attempt）：核心实现文件：新增 RecentPrefillBatchSizeTracker 与估计逻辑，改造 Executor 的 finalize 与 negotiate 接口。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 init_schedule_policy, get_new_batch_prefill）：移除每 pass 的 0.998 衰减，改为用 tracker 在真实 admission 后更新高水位，是行为变更的接入口。
- test/registered/unit/managers/test_prefill_delayer_high_watermark.py（模块 预填延迟器；类别 test；类型 test-coverage；符号 TestPrefillDelayerHighWatermark, test_peak_expires_after_recent_attempt_window, test_recurring_small_peak_remains_effective, test_rejected_non_empty_attempt_advances_window）：新增单元测试，覆盖窗口过期、重复小峰值保持、被拒尝试推进窗口等核心行为，是保障改动正确性的关键。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration；符号 SGLANG_PREFILL_DELAYER_MAX_PREFILL_BS_WINDOW_SIZE）：新增环境变量控制滑动窗口大小，是用户可调参数。
- test/registered/npu/performance/kimi_k2_6/test_npu_kimi_k2_6_w4a8_8p_in3k5_out1k5_20ms.py（模块 性能测试；类别 test；类型 test-coverage）：NPU 性能回归测试，通过设置窗口大小 64 验证 Kimi-K2.6 场景的 TPOT 达标。
- test/registered/npu/performance/minimax_m2_5/test_npu_minimax_m2_5_w8a8_4p_in64k_out1k_prefix90_50ms.py（模块 性能测试；类别 test；类型 test-coverage）：NPU 性能回归测试，同步设置窗口大小 64。
- test/registered/npu/performance/qwen3_6_27b/test_npu_qwen3_6_27b_1p_in1080p_30_out256_50ms.py（模块 性能测试；类别 test；类型 test-coverage）：NPU 性能回归测试，同步设置窗口大小 64。

关键符号：RecentPrefillBatchSizeTracker.init, RecentPrefillBatchSizeTracker.observe_attempt, RecentPrefillBatchSizeTracker.max_prefill_bs, PrefillDelayerSinglePassExecutor.finalize, PrefillDelayerSinglePassExecutor._estimate_attempted_prefill_bs, Scheduler.init_schedule_policy, Scheduler.get_new_batch_prefill

关键源码片段

python/sglang/srt/managers/scheduler.py

移除每 pass 的 0.998 衰减，改为用 tracker 在真实 admission 后更新高水位，是行为变更的接入口。

```
# python/sglang/srt/managers/scheduler.py（节选）
# 初始化调度策略时构建高水位跟踪器，窗口大小由环境变量控制。
self.prefill_bs_tracker = RecentPrefillBatchSizeTracker(
    window_size=envs.SGLANG_PREFILL_DELAYER_MAX_PREFILL_BS_WINDOW_SIZE.get()
)
# max_prefill_bs 从 float 改为 int，避免小数衰减造成的精度丢失。
self.max_prefill_bs: int = 0


def get_new_batch_prefill(self, running_batch: ScheduleBatch) -> NextBatchPlan:
    prefill_delayer_single_pass = None
    if self.prefill_delayer:
        # 不再按调度 pass 衰减，而是等待真实 admission 后更新。
        max_pool_usage = self.pool_stats_observer.get_pool_stats().get_max_pool_usage()
        prefill_delayer_single_pass = PrefillDelayerSinglePassExecutor(
            self.prefill_delayer, token_usage=max_pool_usage
        )

    ret, running_batch = self._get_new_batch_prefill_raw(
        prefill_delayer_single_pass=prefill_delayer_single_pass,
        running_batch=running_batch,
    )

    if self.prefill_delayer:
        # finalize 返回实际准入批大小；若被拒绝则返回估算值（>0），
        # 两者都会进入滑动窗口，使旧峰值在窗口填满后被淘汰。
        observed_prefill_bs = prefill_delayer_single_pass.finalize(
            actual_prefill_bs=ret.batch_size() if ret is not None else 0
        )
        if observed_prefill_bs > 0:
            self.max_prefill_bs = self.prefill_bs_tracker.observe_attempt(
                observed_prefill_bs
            )

    return NextBatchPlan(batch_to_run=ret, running_batch=running_batch)
```

test/registered/unit/managers/test_prefill_delayer_high_watermark.py

新增单元测试，覆盖窗口过期、重复小峰值保持、被拒尝试推进窗口等核心行为，是保障改动正确性的关键。

```
# test/registered/unit/managers/test_prefill_delayer_high_watermark.py
# 核心测试：一个 100 的大峰值在 4 次窗口中被后续小批次淘汰后，
# 高水位应降回当前小批次水平。
def test_peak_expires_after_recent_attempt_window(self):
    tracker = RecentPrefillBatchSizeTracker(window_size=4)

    self.assertEqual(tracker.observe_attempt(100), 100)
```

```
    for attempted_prefill_bs in [2, 1, 2]:
        self.assertEqual(tracker.observe_attempt(attempted_prefill_bs), 100)

    # 窗口满 4 次后，100 被挤出，最大值为当前窗口内的 2。
    self.assertEqual(tracker.observe_attempt(2), 2)


def test_rejected_non_empty_attempt_advances_window(self):
    tracker = RecentPrefillBatchSizeTracker(window_size=4)
    self.assertEqual(tracker.observe_attempt(10), 10)

    # 使用 MagicMock 模拟被拒绝的调度场景：waiting_queue_len=2，
    # 剩余槽位 5，估计批大小为 2。
    delayer = MagicMock()
    delayer.enable_dp_attention = True
    delayer.dp_size = 1
    delayer._metrics_collector = None
    delayer._debug_log_enabled = False
    delayer._negotiate_should_allow_prefill.return_value = _NegotiateOutput(
        next_state=None,
        input_estimation="all",
        output_allow=False,
        output_reason="delay",
        num_prefillable=1,
        num_token_watermark_force_allow=0,
    )

    for _ in range(4):
        executor = PrefillDelayerSinglePassExecutor(delayer, token_usage=0.9)
        self.assertFalse(
            executor.negotiate_should_allow_prefill(
                local_prefillable=True,
                running_batch=15,
                max_prefill_bs=tracker.max_prefill_bs,
                max_running_requests=20,
                waiting_queue_len=2,
            )
        )
        attempted_prefill_bs = executor.finalize(actual_prefill_bs=0)
        self.assertEqual(attempted_prefill_bs, 2)
        tracker.observe_attempt(attempted_prefill_bs)

    # 4 次被拒尝试后，10 被挤出，高水位更新为 2。
    self.assertEqual(tracker.max_prefill_bs, 2)
```

评论区精华

作者 hanwlax 在 issue 评论中向 hanming-lu 请求 review，明确指出这是 #32880 的 follow-up，并解释了原衰减实现导致小高水位快速归零、slot_condition 失效、TPOT 劣化的

现象。评论中强调新方案“保留让过期峰值失效的原始目标，同时把老化过程与 scheduler-pass 频率解耦”。没有实质性的反对意见；hanming-lu 最终批准该 PR。后续评论均为 CI 重跑指令（/rerun-failed-ci），说明主要关注点是 NPU 性能测试的稳定性。

- max_prefill_bs 衰减导致小高水位归零的修复方向 (design): hanming-lu 批准了该方案，认为滑动窗口在保留过期峰值淘汰能力的同时，将老化与 scheduler-pass 频率解耦。

风险与影响

- 风险：
  1. 核心调度热路径变更：get_new_batch_prefill 是每次调度都会执行的路径，新增 finalize 返回值和 tracker 更新虽然是 O(1) 操作，但仍需关注对调度循环延迟的影响，尤其是 DP 场景下的 all_gather 交互。
  2. 估计值可能失真：被拒绝的非空尝试没有真实批大小，_estimate_attempted_prefill_bs 使用 min(等待队列长度，剩余槽位)，在等待队列或槽位变化剧烈时可能高估或低估实际候选批大小，进而影响 slot_condition 的松紧。
  3. 窗口大小敏感：默认窗口 16，但 NPU 性能测试普遍调成 64；不同工作负载下窗口大小直接影响高水位过期速度，默认值可能不是最优。
  4. 类型语义变化：max_prefill_bs 从 float 变为 int，所有相关比较和传参都从浮点退化到整数，需要确认无其他模块依赖其浮点精度。
  5. 回归风险：PrefillDelayerSinglePassExecutor.finalize 的签名从 bool 改为 int，若有其他调用方未同步更新会直接出错；本 PR 已同步所有已知调用点，但需关注未来扩展。 - 影响：对启用 --enable-prefill-delayer 的部署，修复了小 prefill 批场景下 max_prefill_bs 快速归零导致 prefill 干扰 decode、TPOT 恶化的缺陷，直接改善最终用户的首 Token 延迟和输出吞吐；对 NPU 平台影响尤其显著（三个性能测试用例同步调整了窗口大小）。对系统而言，调度器决策逻辑更贴近真实 admission 节奏，行为更可预期。对团队而言，该 PR 提供了新的环境变量和单元测试模板，后续优化调度窗口策略时可复用 RecentPrefillBatchSizeTracker 模式。 - 风险标记：核心调度路径变更，行为语义变化（float->int），估计值可能不准确，窗口大小默认值敏感

关联脉络

- PR #32880 （前作）引入 PrefillDelayer 与 max_prefill_bs 衰减机制：本 PR 是基于该 PR 引入的衰减逻辑的 follow-up，修复其在小批场景下的缺陷。
- PR #34856 fix tpot by adjusting the sliding max-prefill-size window size: 同属 NPU 调度窗口调整以修复 TPOT 的系列工作，虽然时间晚于此 PR，但共享同一调度参数域。