PR #44594 完整报告

vllm-project/vllm

[Core] Add kvcache watermark to reduce preemptions

合并时间：2026-06-11 23:27

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44594

执行摘要

- 一句话：KV cache 水位线减少抢占 82%
- 推荐动作：建议有高并发长输出场景的用户测试此参数；团队可将其作为调度准入控制的参考设计。

功能与动机

在大并发、长输出场景下，请求在解码阶段不断增长，容易耗尽 KV cache 导致大量抢占和重计算，严重拖慢延迟和吞吐。PR 通过预留 KV cache 水位线来吸收这种增长，减少抢占次数。

实现拆解

1. 配置层：在 SchedulerConfig 中新增 watermark 字段（float，[0.0, 1.0)，默认 0.0），通过 EngineArgs 暴露给 CLI。
2. 传递至 KV cache 管理器：从 Scheduler.__init__ 将 watermark 传给 KVCacheManager.__init__，后者计算 watermark_blocks = int(watermark * num_blocks)。
3. 调度侧应用：在 KVCacheManager.allocate_slots() 中增加 has_scheduled_reqs 参数。仅当存在已运行请求且待调度请求状态为 WAITING 或 PREEMPTED 时，在可用块判断中加入 watermark_blocks，从而为新请求保留缓存空间。

关键文件：

- vllm/v1/core/kv_cache_manager.py（模块 缓存管理器；类别 source；类型 core-logic；符号 KVCacheManager.init, KVCacheManager.allocate_slots）：核心变更：添加 watermark 参数并在 allocate_slots 中实现保留逻辑。
- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 Scheduler.init, Scheduler.schedule）：将 watermark 从配置传递到 KVCacheManager，并在 schedule 时传递 has_scheduled_reqs。
- vllm/config/scheduler.py（模块 配置；类别 source；类型 configuration；符号 SchedulerConfig.watermark）：定义 watermark 配置字段及其文档。
- vllm/engine/arg_utils.py（模块 参数解析；类别 source；类型 configuration；符号 EngineArgs.watermark, EngineArgs.add_cli_args, EngineArgs.create_engine_config）：将 watermark 暴露为 CLI 参数 --watermark，并传递到 SchedulerConfig。

- benchmarks/kv_cache_watermark.sh（模块 基准测试；类别 other；类型 test-script；符号 g）：新增基准脚本用于复现和验证 watermark 效果。
- tests/v1/core/test_scheduler.py（模块 测试；类别 test；类型 test-coverage；符号 create_scheduler_with_priority）：在测试中显式设置 watermark=0.0 以保持确定性行为。
- tests/v1/core/utils.py（模块 测试；类别 test；类型 test-coverage）：在测试工具中设置 watermark=0.0 以确保一致性。

关键符号：KVCacheManager.init, KVCacheManager.allocate_slots, Scheduler.init, Scheduler.schedule, SchedulerConfig.init

关键源码片段

vllm/v1/core/kv_cache_manager.py

核心变更：添加 watermark 参数并在 allocate_slots 中实现保留逻辑。

```
class KVCacheManager:
    def __init__(
        self,
        # ... 其他参数 ...
        watermark: float = 0.0, # 新增：水位线比例
    ) -> None:
        # ... 初始化其他属性 ...
        self.watermark_blocks = int(watermark * kv_cache_config.num_blocks) # 计算保留块数

    def allocate_slots(
        self,
        request,
        # ... 其他参数 ...
        has_scheduled_reqs: bool = True, # 新增：是否有已调度请求
    ) -> KVCacheBlocks | None:
        watermark_blocks = 0
        # 仅对等待或抢占请求且已有运行请求时应用水位线
        if has_scheduled_reqs and request.status in (
            RequestStatus.WAITING,
            RequestStatus.PREEMPTED,
        ):
            watermark_blocks = self.watermark_blocks

        # 在完整序列长度检查中加入水位线
        if full_sequence_must_fit:
            # ... 计算 num_blocks_to_allocate ...
            required_blocks = num_blocks_to_allocate + watermark_blocks
            if required_blocks > self.block_pool.get_num_free_blocks():
                return None

        # 在普通分配检查中加入水位线（与 reserved_blocks 类似）
        available_blocks = self.block_pool.get_num_free_blocks() - reserved_blocks
        required_blocks = num_blocks_to_allocate + watermark_blocks
```

```
    if required_blocks > available_blocks:
        return None
    # ... 分配逻辑 ...
```

评论区精华

WoosukKwon 建议 watermark 应考虑运行中的请求数量，njhill 先尝试按请求数缩放并设上限（4%），但最终改回简单百分比（commit ca66c1d），理由是简单可预测。合并版本使用总块数的固定比例，默认 0 禁用。

- Watermark 参数设计：从按请求数缩放改为简单百分比 (design): 采用总 KV cache 块数的固定比例作为 watermark，默认 0 禁用。

风险与影响

- 风险：开启 watermark 会保留部分 KV cache，在非饱和负载下可能降低资源利用率；但 benchmark 显示在高负载下整体吞吐和延迟均有改善。当前默认禁用，无副作用。与 scheduler_reserve_full_isl 共存时可能需要调优。
- 影响：
  - 用户：新增 --watermark 参数，可在高并发长输出场景下稳定减少抢占，提升服务质量。
  - 系统：核心调度准入逻辑变更，与其他调度策略（如优先级、分块预填充）交互需持续关注。
  - 团队：新增 benchmark 脚本（benchmarks/kv_cache_watermark.sh）便于复现和调优。
  - 风险标记：默认禁用，新增配置参数，核心调度路径变更

关联脉络

- PR #44560 [Bugfix] Fix async KV-load deadlock: 该 PR 的改动与 #44594 在 allocate_slots 的 reserved_blocks 参数上存在冲突，njhill 在合并时解决了冲突（commit de74451）。