PR #40946 完整报告

vllm-project/vllm

[Bugfix] Cap SWA/chunked-local runtime admission to startup pool-sizing bound

合并时间：2026-04-27 12:33

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40946

执行摘要

- 一句话：修复 SWA 运行时准入与启动池大小计算不一致导致调度死锁
- 推荐动作：建议仔细阅读此 PR，尤其是单真源设计（singlesource of truth）在 admission 与 pool sizing 间的应用。值得关注的设计决策：将 cap 逻辑置于 manager 内部而非 coordinator 层，避免了 #39866/#40027 中的准入与分配分裂问题。代码实现简洁，注释清晰，适合作为 KV cache 准入逻辑的学习参考。

功能与动机

修复 issue #39734：调度器死锁避免。混合全注意力和滑动窗口模型中，启动池大小基于回收边界计算，但运行时准入门控未考虑回收，导致长请求（如 16K token）被拒绝，而池中仍有空闲块。此补丁还解决了 #39866/#40027 的根本原因。

实现拆解

1. 在 vllm/v1/kv_cache_interface.py 中为 SlidingWindowSpec 和 ChunkedLocalAttentionSpec 新增 max_admission_blocks_per_request 方法，该方法返回每个请求在考虑回收情况下的最大块数，并将 max_memory_usage_bytes 重构为调用该方法，从而统一启动池大小和运行时准入的上限。
2. 修改 vllm/v1/core/single_type_kv_cache_manager.py 中的 SingleTypeKVCacheManager，使其构造函数接受可选的 max_admission_blocks_per_request 参数，并在 get_num_blocks_to_allocate 中用它来限制 num_required_blocks。只有 SWA 和 ChunkedLocalAttention 类型的 manager 才会传递该上限，其他类型（如 full attention）不受影响。
3. 在 get_manager_for_kv_cache_spec 工厂函数中，根据 spec 类型自动计算 max_admission_blocks_per_request 并传递给 manager，避免调用方手动处理。
4. 通过 vllm/v1/core/kv_cache_coordinator.py、vllm/v1/core/kv_cache_manager.py、vllm/v1/core/sched/scheduler.py 和 vllm/v1/simple_kv_offload/manager.py 传递新增的 max_num_batched_tokens 参数，使得 spec 方法能在 manager 构造时正确计算上限。默认值设为 max_model_len 以保持向后兼容。
5. 在 tests/v1/core/test_prefix_caching.py 中新增两个测试用例：验证 SWA 上限生效后长请求能被准入，并确认 full attention 仍能正确拒绝过大的请求。

关键文件：

- vllm/v1/kv_cache_interface.py（模块 缓存接口；类别 source；类型 core-logic；符号 max_memory_usage_bytes, max_admission_blocks_per_request）：核心设计文件：新增 max_admission_blocks_per_request 方法，重构 max_memory_usage_bytes 以调用该方法，实现单真源统一准入上限。
- tests/v1/core/test_prefix_caching.py（模块 前缀缓存测试；类别 test；类型 test-coverage；符号 test_can_fit_full_sequence_swa_cap_admits_long_prompt, test_can_fit_full_sequence_full_attention_still_gates_oversized）：新增两个测试用例，覆盖 SWA 上限生效后长请求准入及 full attention 仍能拒绝过大请求的关键场景。
- vllm/v1/core/single_type_kv_cache_manager.py（模块 单类型管理；类别 source；类型 core-logic）：实现运行时准入上限的核心逻辑：构造函数接收 max_admission_blocks_per_request，get_num_blocks_to_allocate 中使用该上限限制请求块数。
- vllm/v1/core/kv_cache_coordinator.py（模块 协调器；类别 source；类型 core-logic）：传递 max_num_batched_tokens 参数到 SingleTypeKVCacheManager 构造函数，支持自动计算准入上限。
- vllm/v1/core/kv_cache_manager.py（模块 管理器；类别 source；类型 core-logic）：传递 max_num_batched_tokens 参数到 KVCacheCoordinator，默认 fallback 为 max_model_len。
- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic）：传递 max_num_batched_tokens 参数到 KVCacheManager，确保运行时准入上限可用。
- vllm/v1/simple_kv_offload/manager.py（模块 卸载管理器；类别 source；类型 core-logic）：传递 max_num_batched_tokens 参数到 offload 路径的 coordinator，保持一致性。
- tests/v1/core/test_single_type_kv_cache_manager.py（模块 单类型测试；类别 test；类型 test-coverage）：更新测试脚手架以支持新的 max_num_batched_tokens 参数。

关键符号：max_admission_blocks_per_request, get_num_blocks_to_allocate, get_manager_for_kv_cache_spec, test_can_fit_full_sequence_swa_cap_admits_long_prompt, test_can_fit_full_sequence_full_attention_still_gates_oversized

关键源码片段

vllm/v1/kv_cache_interface.py

核心设计文件：新增 max_admission_blocks_per_request 方法，重构 max_memory_usage_bytes 以调用该方法，实现单真源统一准入上限。

```
# vllm/v1/kv_cache_interface.py（重构后的关键方法）

class SlidingWindowSpec(AttentionSpec):
    # ... 其他属性 ...

    def max_admission_blocks_per_request(
        self, max_num_batched_tokens: int, max_model_len: int
    ) -> int:
        """每个请求的准入上限，单位块。
```

```
        启动池大小 (`max_memory_usage_bytes`) 和运行时准入门控都使用此上限，
        确保两者一致，避免 #39734 中的死锁。
        """
        # 在分块 prefill 期间，我们最多持有最后一个 `sliding_window-1`
        # 个已计算 token 加上新调度的 token，且不超过 `max_model_len`。
        num_tokens = min(
            self.sliding_window - 1 + max_num_batched_tokens, max_model_len
        )
        # +1 因为滑动窗口可能不从块起始位置对齐。
        return cdiv(num_tokens, self.block_size) + 1

    def max_memory_usage_bytes(self, vllm_config: VllmConfig) -> int:
        # 委托给单真源方法，保持公式一致。
        max_blocks = self.max_admission_blocks_per_request(
            max_num_batched_tokens=vllm_config.scheduler_config.max_num_batched_tokens,
            max_model_len=vllm_config.model_config.max_model_len,
        )
        return max_blocks * self.page_size_bytes

class ChunkedLocalAttentionSpec(AttentionSpec):
    # ... 其他属性 ...

    def max_admission_blocks_per_request(
        self, max_num_batched_tokens: int, max_model_len: int
    ) -> int:
        # 分块 prefill 期间，最多持有 attention_chunk_size + 一批新 token
        num_tokens = min(
            self.attention_chunk_size + max_num_batched_tokens, max_model_len
        )
        return cdiv(num_tokens, self.block_size)

    def max_memory_usage_bytes(self, vllm_config: VllmConfig) -> int:
        max_blocks = self.max_admission_blocks_per_request(
            max_num_batched_tokens=vllm_config.scheduler_config.max_num_batched_tokens,
            max_model_len=vllm_config.model_config.max_model_len,
        )
        return max_blocks * self.page_size_bytes
```

vllm/v1/core/single_type_kv_cache_manager.py

实现运行时准入上限的核心逻辑：构造函数接收 max_admission_blocks_per_request，get_num_blocks_to_allocate 中使用该上限限制请求块数。

```
# vllm/v1/core/single_type_kv_cache_manager.py ( 构造和准入 cap)

class SingleTypeKVCacheManager(ABC):
    def __init__(
        self,
        kv_cache_spec: KVCacheSpec,
        block_pool: BlockPool,
```

```
        enable_caching: bool,
        kv_cache_group_id: int,
        dcp_world_size: int = 1,
        pcp_world_size: int = 1,
        max_admission_blocks_per_request: int | None = None, # 新增
    ) -> None:
        # ... 其他初始化 ...
        if dcp_world_size * pcp_world_size > 1:
            self.block_size *= dcp_world_size * pcp_world_size
        # ...
        self._max_admission_blocks_per_request = max_admission_blocks_per_request
        # ...

    def get_num_blocks_to_allocate(self, ...) -> int:
        """
        获取需要分配的块数。对于 SWA/ChunkedLocalAttention,
        使用上限确保准入门控与启动池大小一致。
        """
        num_required_blocks = cdiv(num_tokens, self.block_size)
        # 如果设置了上限，则限制请求所需的块数（回收感知）。
        if self._max_admission_blocks_per_request is not None:
            num_required_blocks = min(
                num_required_blocks, self._max_admission_blocks_per_request
            )
        num_req_blocks = len(self.req_to_blocks.get(request_id, ()))
        # ... 后续计算 ...


def get_manager_for_kv_cache_spec(
    kv_cache_spec: KVCacheSpec,
    max_num_batched_tokens: int,
    max_model_len: int,
    **kwargs,
) -> SingleTypeKVCacheManager:
    manager_class = spec_manager_map[type(kv_cache_spec)]
    # 对于循环回收块的 spec，自动计算准入上限。
    if isinstance(kv_cache_spec, (SlidingWindowSpec, ChunkedLocalAttentionSpec)):
        kwargs["max_admission_blocks_per_request"] = (
            kv_cache_spec.max_admission_blocks_per_request(
                max_num_batched_tokens=max_num_batched_tokens,
                max_model_len=max_model_len,
            )
        )
    manager = manager_class(kv_cache_spec, **kwargs)
    return manager
```

评论区精华

Review 中主要讨论来自 njhill，他提供了简化建议的 commit（f0c7ac64），并最终批准了 PR。Gemini-code-assist 自动审核认为代码变更无问题。PR 作者在 issue 评论中确认不需要 #39866 因为该 PR 在 coordinator 层做限制，而本 PR 在 manager 层统一处理，设计更优。

- 简化变更实现 (design): Dao 接受了建议，njhill 直接提交了简化 commit 到分支，最终合并。

风险与影响

- 风险：
  1. 核心准入路径变更：修改了 get_num_blocks_to_allocate 和 max_memory_usage_bytes 两个核心方法，任何回归都可能导致准入错误或 OOM。
  2. 参数传递遗漏：新增 max_num_batched_tokens 参数需要传递到多个 constructor（coordinator, manager, scheduler, offload manager），若某处传递错误或默认值未生效，可能导致上限计算不正确。
  3. 向后兼容：默认值 max_model_len 保证了未传入 max_num_batched_tokens 时行为与之前一致，但需要验证所有调用点均已更新。
  4. 测试覆盖：虽然新增了测试用例，但未覆盖 ChunkedLocalAttention 的类似场景，且未覆盖 offload 路径。 - 影响：影响范围：SWA 和 ChunkedLocalAttention 的准入逻辑，涉及 Gemma4 等混合注意力模型。影响程度：修复调度死锁，提高并发吞吐（PR 中报告 Gemma3 12B 上从 4 请求等待变为全部通过）。对纯 full attention 模型无影响。团队需关注参数传递完整性。 - 风险标记：核心准入路径变更，参数传递遗漏风险，启动与运行时一致性依赖

关联脉络

- PR #39866 [Scheduler] Cap SWA admission budget at sliding_window + chunk_size: 解决相同症状但设计不同（在 coordinator 层 cap）；本 PR 采用更统一的经理层方案，避免了 #39866 的准入 / 分配分裂问题。
- PR #40027 Handle capped SWA admission in allocator path: 跟随 #39866 的补丁，处理准入与分配不匹配；本 PR 的设计消除了这种不匹配，因此不需要 #40027。
- PR #39734 [Bug]: Scheduler deadlocks when request exceeds KV cache capacity but is within max_model_len: 本 PR 直接修复的根本 bug：运行时准入门控与启动池大小计算不一致导致调度死锁。