PR #51725 完整报告

vllm-project/vllm

[Perf] Adaptive budget for spec scheduled token, 55%~65% E2E TTFT Improvement

合并时间：2026-08-11 11:52

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51725

执行摘要

- 一句话：投机解码调度预算自适应，低并发 TTFT 最多降 65%
- 推荐动作：值得精读：这是一次小改动（+54/-24）带来大收益（TTFT -55%~-65%）的典型性能优化，核心思想是把「最坏情况预留」改为「按实际调度动态记账」。重点关注 schedule() 中 input_budget 与 draft_slots 的四个记账点（RUNNING/WAITING/ 抢占归还 / 新请求）以及 vllm.py 中校验语义的放宽。若团队也维护投机解码部署，建议合入后在各自的注意力后端与模型组合上做一轮回归，并关注后续 benchislett 的可能 follow-on 意见。

功能与动机

旧逻辑为投机解码预留草稿槽位时按最大并发数扣减：max_num_seqs=1024、max_num_batched_tokens=8192 时每步只能调度 2048 个 token，而请求数只有 1/32/128 时绝大多数预留槽位闲置。PR body 明确说明：This PR make the scheduled tokens much larger when request num is small by using an adaptive strategy. This is extremely helpful when request num is small and give very huge perf improvement。作者以 Kimi-K3 + DSpark 投机 + FLASHINFER_MLA 实测 1/4/16 并发下 TTFT 分别改善 65.3% / 55.5% / 62.2%。

实现拆解

1. 配置层：撤销全局预留（vllm/config/vllm.py）：_set_max_num_scheduled_tokens() 不再用 max_num_new_slots_for_drafting * max_num_seqs 扣减调度预算，默认 max_num_scheduled_tokens 直接等于 max_num_batched_tokens；校验从「必须容纳全部请求的草稿槽位」放宽为「必须大于单个请求的草稿槽位数」。这是整个自适应的前提——预算不再被最大并发数绑架。
2. 调度器：按实际调度请求记账（vllm/v1/core/sched/scheduler.py）：schedule() 新增 draft_slots（每请求草稿槽位数）与 input_budget（模型每步总输入上限）。RUNNING 与 WAITING 两个调度循环统一用 min(token_budget, input_budget - draft_slots) 计算可调度 token 数，每次成功调度后 input_budget -= num_new_tokens + draft_slots；剩余预算不足一个请求的草稿槽位时提前 break，草稿槽位只在请求真正被调度时才占用。
3. 抢占与断言配套：抢占回退路径同步归还预算（token_budget += restored、input_budget += restored + draft_slots）；新增 assert input_budget >= 0 兜底，与既有 total_num_scheduled_tokens <= max_num_scheduled_tokens 断言共同守住不变量。

4. 测试配套（tests/v1/core/utils.py、tests/v1/core/test_scheduler.py）：create_scheduler 新增 parallel_drafting 参数透传；新增 test_draft_slots_budgeted_per_scheduled_request，用 max_num_batched_tokens=20、max_num_seqs=16、num_speculative_tokens=4（每请求 3 个草稿槽位）验证两个 10-token 请求分别得到 10/4 个调度 token——旧逻辑下该配置会因预算为负直接报错。

关键文件：

- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 Scheduler.schedule）：核心调度循环的预算记账重构：引入 draft_slots 与 input_budget，草稿槽位从按 max_num_seqs 全局预留改为按实际调度请求占用，是本 PR 收益的根源。
- vllm/config/vllm.py（模块 配置层；类别 source；类型 core-logic；符号 VllmConfig._set_max_num_scheduled_tokens）：改变投机解码下 max_num_scheduled_tokens 的默认推导与校验语义，是自适应预算生效的前提。
- tests/v1/core/test_scheduler.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_draft_slots_budgeted_per_scheduled_request）：新增针对 per-request 草稿槽位记账的单元测试，是唯一直接验证新预算语义的测试入口。
- tests/v1/core/utils.py（模块 测试基建；类别 test；类型 test-coverage；符号 create_scheduler）：create_scheduler 新增 parallel_drafting 参数透传，支撑新测试构造 parallel drafting 配置。

关键符号：Scheduler.schedule, VllmConfig._set_max_num_scheduled_tokens, create_scheduler, test_draft_slots_budgeted_per_scheduled_request

关键源码片段

vllm/config/vllm.py

改变投机解码下 max_num_scheduled_tokens 的默认推导与校验语义，是自适应预算生效的前提。

```
def _set_max_num_scheduled_tokens(self):
    """
    设置调度器每步可调度的 token 上限。

    旧实现：只要开启投机解码，就按「每请求草稿槽位数 × max_num_seqs」
    提前从 max_num_batched_tokens 中扣除草稿占位，得到 max_num_scheduled_tokens。
    以 max_num_seqs=1024、每请求 6 个草稿槽位为例，默认 8192 的预算会被压到
    2048，即使当前只有 1 个请求在跑。

    新实现：不再按最大并发数全局预留，默认调度预算直接等于
    max_num_batched_tokens；草稿槽位改由调度器在 schedule() 中按
    「实际被调度请求数」动态记账。
    """
    if self.speculative_config is not None:
        scheduled_token_delta = self.speculative_config.max_num_new_slots_for_drafting
        max_num_batched_tokens = self.scheduler_config.max_num_batched_tokens
        if self.scheduler_config.max_num_scheduled_tokens is None:
```

```
            self.scheduler_config.max_num_scheduled_tokens = max_num_batched_tokens

        if self.scheduler_config.max_num_scheduled_tokens <= 0:
            raise ValueError(
                "max_num_scheduled_tokens is set to"
                f" {self.scheduler_config.max_num_scheduled_tokens} based on"
                " the speculative decoding settings, which does not allow"
                " any tokens to be scheduled. Increase max_num_batched_tokens"
                " to accommodate the additional draft token slots, or decrease"
                " num_speculative_tokens."
            )

        # 只需保证 batch 预算能容纳单个请求的草稿槽位即可，
        # 不再要求容纳 max_num_seqs 个请求的草稿槽位总和。
        if max_num_batched_tokens <= scheduled_token_delta:
            raise ValueError(
                "VllmConfig does not have enough slots to schedule a token and"
                " support the speculative decoding settings."
                f" Got {max_num_batched_tokens=} and {scheduled_token_delta=}."
            )
```

tests/v1/core/test_scheduler.py

新增针对 per-request 草稿槽位记账的单元测试，是唯一直接验证新预算语义的测试入口。

```
def test_draft_slots_budgeted_per_scheduled_request(tmp_path, monkeypatch):
    """验证草稿槽位按「实际被调度的请求」记账，而不是按 max_num_seqs 预留。"""
    monkeypatch.setenv("VLLM_USE_V2_MODEL_RUNNER", "0")
    (tmp_path / "config.json").write_text(
        '{"architectures": ["OPTForCausalLM"], "model_type": "opt"}'
    )
    # 用极小的输入预算放大记账效果：max_num_batched_tokens=20，
    # 每请求草稿槽位 3 个（num_speculative_tokens=4 时 max_num_new_slots_for_drafting=3）。
    scheduler = create_scheduler(
        model=str(tmp_path),
        max_num_seqs=16,
        max_num_batched_tokens=20,
        num_speculative_tokens=4,
        parallel_drafting=True,
        skip_tokenizer_init=True,
    )
    speculative_config = scheduler.vllm_config.speculative_config
    assert speculative_config is not None
    # 旧逻辑下 max_num_scheduled_tokens = 20 - 3*16 < 0 会直接抛错；
    # 新逻辑默认调度预算等于 max_num_batched_tokens。
    assert scheduler.max_num_scheduled_tokens == 20
    assert speculative_config.max_num_new_slots_for_drafting == 3

    for request in create_requests(num_requests=2, num_tokens=10):
        scheduler.add_request(request)
```

```
# 请求 0 拿走 10 个 token（剩余输入预算 20-10-3=7）；
# 请求 1 最多拿到 7-3=4 个 token。
assert scheduler.schedule().num_scheduled_tokens == {"0": 10, "1": 4}
```

评论区精华

评审几乎全部集中在 njhill 的 Approve 评论，diff 上没有行级 review 评论。njhill 认可实现「very clean」，并说明本希望同时拿到 benchislett 的确认，但考虑到上线时效已先行合并，后续可依 benchislett 意见做 follow-on 修改。主要风险背书来自作者给出的 benchmark（对应 #51726）与三轮 CI（#83199 / #83207 / #83264）。

- 合并审批与二次评审意见 (design): 变更整体获认可，出于上线时效先行合并，允许在 benchislett 后续反馈时做 follow-on 修改。

风险与影响

- 风险： 1) 核心调度路径变更：schedule() 是每步执行的热路径，本次在 RUNNING/WAITING/ 抢占回退 / 新请求四个分支同时引入 input_budget 记账，任何分支漏记或重复归还都会导致预算漂移；新增 assert input_budget >= 0 能兜底负值，但不会纠正提前 break 造成的调度偏差。 2) 配置校验语义放宽：旧逻辑拒绝「max_num_batched_tokens 容不下 max_num_seqs 个请求草稿槽位」的配置，新逻辑只要求大于单个请求的槽位数，此前报错的配置现在会通过并进入实际调度，行为差异需要留意。 3) 跨后端验证有限：benchmark 只覆盖 Kimi-K3 + FLASHINFER_MLA + parallel drafting 场景，其他注意力后端（如 Triton MLA）、串行投机、多步调度与 CUDA graph 填充下的 budget 行为未在 PR 内验证。 4) 关联系统耦合：max_num_scheduled_tokens 语义变化会影响依赖它的其他模块（如 CUDA graph capture 尺寸、prefill workspace 估算等），需在后续 CI 与发布中观察。
  - 影响：对用户：所有启用投机解码的部署每步可调度 token 数都会变化——低并发交互式场景（1~16 并发）收益最大，TTFT 缩短逾半；高并发场景与旧逻辑收敛。对系统：调度器预算模型更贴近真实占用，理论上能更充分利用 max_num_batched_tokens 的算力，但也要求 input_budget 记账在所有调度分支精确一致。对团队：该模式为「每请求草稿槽位动态记账」树立了先例，后续投机解码相关调度改动（如多草稿、动态投机数）都应沿用同一预算口径；合并节奏上演示了「先合并拿收益、再补二次评审」的决策。
  - 风险标记：核心调度路径变更，配置校验语义放宽，跨后端覆盖待验证，预算记账一致性风险

关联脉络

- PR #51726 （PR body 中引用的配套 benchmark 结果 PR）：作者在 PR body 中明确说明将 benchmark 结果与实现拆成两个 PR（benchmark result combined #51726），便于独立 review 与 revert。
- PR #46849 [MRV2][Spec] Fuse AR speculator multi-step decodes back into one CUDA graph: 同属 v1 投机解码执行路径的性能优化线，涉及 spec 多步 decode 调度与 CUDA graph 融合，与本 PR 的调度预算改动在同一执行路径上相互影响。

- PR #50713 [CI] Solidify speculative decoding E2E coverage: 投机解码 E2E 测试基建，覆盖调度语义变更的回归验证。