PR #35382 完整报告

sgl-project/sglang

[Refactor] Share the page-aligned decode alloc lens between EAGLE and DFLASH

合并时间：2026-08-19 05:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35382

执行摘要

- 一句话：抽取 EAGLE 与 DFLASH 共用的页对齐 *decode* 分配水位计算
- 推荐动作：值得快速精读，特别是 page_aligned_decode_alloc_lens 的封装方式：它把 " 页对齐 + max(cur, ...) 钳制 + 记账 " 三个关注点打包成宿主侧纯计算函数，是理解 SGLang 分配器从设备回读走向宿主记账的关键一环。对正在跟进 #35223 的读者，这篇 PR 提供了一个清晰的去重范本；对一般读者，可结合 #35265 对比 " 先页对齐、再共享 " 的演进时序。

功能与动机

PR body 说明：EAGLE 和 DFLASH 在 *spec-v2 decode* 中重复实现 (cur, nxt) watermark 计算，需要提取到 *allocation_sizing.page_aligned_decode_alloc_lens*。tracking issue #35223 的 in-flight 清单中明确指出：本 PR 的目标是 *dedup*，让 watermark 循环成为 kv_allocated_len 的唯一写者，服务于 " 宿主侧记账 + 页对齐分配 " 的整体设计——分配在宿主侧按整页规划，*free* 不再依赖设备回读。

实现拆解

本 PR 是一步跨模块去重重构，按以下步骤实施：

1. 新增共享函数（allocation_sizing.py）：在 *get_alloc_reserve_per_decode*() 之后新增 *page_aligned_decode_alloc_lens*(reqs, *, *reserve*, *page_size*)。函数逐请求读取 r.kv.kv_allocated_len 作为 *cur*，计算 nxt = max(cur, *ceil*((r.kv_committed_len + *reserve*) / *page_size*) * *page_size*)，并累加 num_needed_tokens = nxt - cur。该计算完全发生在宿主侧，不触碰设备张量，符合 #35223 的宿主侧记账方向；max(cur, ...) 保留了 EAGLE 侧针对 *adaptive* downswitch 的 "nxt 不回落 " 钳制。
2. EAGLE 侧接线（*eagle_utils.py*）：*eagle_prepare_for_decode*() 删除内联的 cur_kv_lens/nxt_kv_lens 循环，改为以 *reserve=double_alloc*（即 *get_alloc_reserve_per_decode*()）调用共享函数；原先在循环内执行的 r.*decode_batch_idx* += 1 被移到调用后的独立 for 循环，语义不变。共享函数返回的普通列表仍由原逻辑转换为 CPU 张量、再做 non-blocking H2D 与 row-width 断言。
3. DFLASH 侧接线（dflash_info_v2.py）：DFlashDraftInputV2.*prepare_for_decode*() 删除内联循环，改为 *reserve* = 2 * block_size（block_size = *speculative_num_draft_tokens*）传入共享函数；原先在循环内兼做的 top_k 统计（

max_top_k、uniform_top_k、nxt_kv_lens_sum、committed_seq_lens_sum）保留在调用后的独立循环中，nxt_kv_lens_cpu_t 的切片声明提前到 buffer 准备区。

4. 配套与验证：无新增测试文件；作者对三个 spec 测试文件执行 /rerun-test，在 1-gpu-5090 上全部通过。

下表对照两个调用方的差异保留情况：

调用方	reserve 来源	保留的特有逻辑
EAGLE	double_alloc = 2 * get_alloc_len_per_decode()	row-width 断言、decode_batch_idx += 1
DFLASH	2 * speculative_num_draft_tokens	max_top_k/uniform_top_k 统计、verify block 校验

关键文件：

- python/sglang/srt/mem_cache/allocation_sizing.py（模块 分配计算；类别 source；类型 core-logic；符号 page_aligned_decode_alloc_lens）：本 PR 的核心：新增共享函数 page_aligned_decode_alloc_lens，集中承载 EAGLE 与 DFLASH 共用的页对齐 decode 分配水位计算，是 #35223 宿主记账路线中 kv_allocated_len 单一写者的雏形。
- python/sglang/srt/speculative/dflash_info_v2.py（模块 投机解码；类别 source；类型 dependency-wiring；符号 DFlashDraftInputV2.prepare_for_decode）：DFLASH 侧接线：prepare_for_decode 删除内联 watermark 循环，改用共享函数，保留 top_k 统计等 DFLASH 特有逻辑。
- python/sglang/srt/speculative/eagle_utils.py（模块 投机解码；类别 source；类型 dependency-wiring；符号 eagle_prepare_for_decode）：EAGLE 侧接线：eagle_prepare_for_decode 删除重复的 watermark 循环，改为共享函数调用，decode_batch_idx 递增移到独立循环。

关键符号：page_aligned_decode_alloc_lens, eagle_prepare_for_decode, DFlashDraftInputV2.prepare_for_decode

关键源码片段

python/sglang/srt/mem_cache/allocation_sizing.py

本 PR 的核心：新增共享函数 page_aligned_decode_alloc_lens，集中承载 EAGLE 与 DFLASH 共用的页对齐 decode 分配水位计算，是 #35223 宿主记账路线中 kv_allocated_len 单一写者的雏形。

```
def page_aligned_decode_alloc_lens(
    reqs,
    *,
    reserve: int,
    page_size: int,
):
    """Whole-page decode alloc lens: nxt rounds committed up to page so allocated
```

```
    == recorded (unaligned tails leak at ps>1).

    EAGLE 与 DFLASH 共用这段 watermark 计算：以请求当前的
    kv_allocated_len 为下界，把 kv_committed_len + reserve 整体向上取整到
    页边界，避免"已分配但未记账"的尾部泄漏。全部在宿主侧完成，
    不触碰设备张量，返回值由调用方决定何时写入 CPU/GPU buffer。
    """
    cur_kv_lens = [0] * len(reqs)
    nxt_kv_lens = [0] * len(reqs)
    num_needed_tokens = 0
    for i, r in enumerate(reqs):
        # cur 是请求对象上的宿主侧分配水位，本函数是它的单一写者入口
        cur = r.kv.kv_allocated_len
        # max(cur, ...) 保证 adaptive downswitch 时 nxt 不会回落
        nxt = max(
            cur,
            (r.kv_committed_len + reserve + page_size - 1) // page_size * page_size,
        )
        cur_kv_lens[i] = cur
        nxt_kv_lens[i] = nxt
        num_needed_tokens += nxt - cur
    return cur_kv_lens, nxt_kv_lens, num_needed_tokens
```

python/sglang/srt/speculative/dflash_info_v2.py

DFLASH 侧接线：prepare_for_decode 删除内联 watermark 循环，改用共享函数，保留 top_k 统计等 DFLASH 特有逻辑。

```
    def prepare_for_decode(self, batch: ScheduleBatch):
        """为下一个 DFLASH 解码步在共享 req_to_token 池中预留 headroom。

        DFLASH spec-v2 使用 overlap 调度的 over-allocation 策略：提前预留未来
        KV 槽位，worker 可直接从 req_to_token 收集 out_cache_loc，
        免去 allocator 的 backup/restore。
        """
        plan_stream, plan_stream_ctx = _get_overlap_plan_stream(batch.device)
        bs = batch.batch_size()
        if bs == 0:
            return
        batch.maybe_evict_swa()
        self._ensure_prepare_length_buffers(bs, batch.device)
        # 各 CPU/GPU 长度缓冲必须已就绪，后续非阻塞 copy 依赖它们
        assert self._prepare_batch_seq_lens_cpu_buf is not None
        assert self._prepare_cur_kv_lens_cpu_buf is not None
        assert self._prepare_nxt_kv_lens_cpu_buf is not None
        # ... GPU 缓冲断言在 head 版本中同样保留
        batch_seq_lens_cpu_t = self._prepare_batch_seq_lens_cpu_buf[:bs]
        cur_kv_lens_cpu_t = self._prepare_cur_kv_lens_cpu_buf[:bs]
        nxt_kv_lens_cpu_t = self._prepare_nxt_kv_lens_cpu_buf[:bs]
```

```
# DFLASH 每个解码步需要固定大小的 verify block
block_size = int(get_spec().speculative_num_draft_tokens)
if block_size <= 0:
    raise ValueError(f"DFLASH invalid speculative_num_draft_tokens={block_size}.")
reserve = 2 * block_size # 与 EAGLE 的双缓冲语义对齐，吸收 kv_committed_len 延迟
page_size = batch.token_to_kv_pool_allocator.page_size

# 与 EAGLE 共享同一套页对齐 watermark 计算（本 PR 的去重点）
cur_kv_lens, nxt_kv_lens, num_needed_tokens = page_aligned_decode_alloc_lens(
    batch.reqs,
    reserve=reserve,
    page_size=page_size,
)

# DFLASH 特有：top_k 统计，用于后续 verify 规划
max_top_k = 1
uniform_top_k_value = None
uniform_top_k = True
nxt_kv_lens_sum = 0
committed_seq_lens_sum = 0
for i, (req, cur, nxt) in enumerate(zip(batch.reqs, cur_kv_lens, nxt_kv_lens)):
    committed_len = int(req.kv_committed_len)
    committed_seq_lens_sum += committed_len
    top_k = int(req.sampling_params.top_k)
    batch_seq_lens_cpu_t[i] = committed_len
    cur_kv_lens_cpu_t[i] = cur
    nxt_kv_lens_cpu_t[i] = nxt
    nxt_kv_lens_sum += nxt
    # 更新 max_top_k 与 uniform_top_k 判定
    if top_k > max_top_k:
        max_top_k = top_k
    if i == 0:
        uniform_top_k_value = top_k
    elif uniform_top_k and top_k != uniform_top_k_value:
        uniform_top_k = False

self.max_top_k = max(max_top_k, 1)
self.uniform_top_k_value = uniform_top_k_value if uniform_top_k else None
# ... 后续 plan_stream 上下文与 alloc_for_spec_decode 不变
```

评论区精华

本 PR 没有出现代码 review 评论（review_comments_count = 0），仅存在 3 条 CI 控制类评论，无实质设计交锋：

> hnyls2002: /rerun-test test/registered/spec/eagle/test_spec_eagle_topk.py test/registered/spec/dflash/test_dflash.py test/registered/unit/spec/test_dflash_overlap_hostsync.py

> github-actions[bot]: Results for /rerun-test ... 1-gpu-5090 (3 tests): 😕

结论：作者以现有 spec 回归测试兜底行为等价性，未对共享函数新增独立单测。

- CI rerun 与回归范围 (other): 1-gpu-5090 上三个测试全部通过（😕），无失败。

风险与影响

- 风险：
  1. 行为等价性风险（低）：EAGLE 侧原公式为 max(cur, (kv_committed_len + double_alloc + page_size - 1) // page_size * page_size)，DFLASH 侧原公式为 max(cur, (committed_len + 2 * block_size + page_size - 1) // page_size * page_size)，替换后完全一致。唯一细微差别是 DFLASH 原先在循环内对 kv_committed_len 做 int()，共享函数直接用原值参与整数运算，结果等价。
  2. 测试覆盖风险（中）：这是核心分配路径的重构，但未新增针对 page_aligned_decode_alloc_lens 的独立单测；行为等价性完全依赖现有 spec 测试。若后续调整 reserve 语义，缺少精确的断言锚点。
  3. 调用方语义耦合风险（低 - 中）：共享函数签名中 reserve 只传整数，不表达 " 双缓冲 / 2x draft tokens" 的业务语义。EAGLE 与 DFLASH 之所以各传各的值，依赖调用方对 get_alloc_reserve_per_decode() 和 2 * block_size 的理解，未来新调用方可能误用。
  4. 关键路径影响（低）：spec-v2 decode 的 KV 预留处于吞吐关键路径，但本次为纯计算提取，不改变分配大小、时机，也不新增设备同步或 stream 操作，无性能回归预期。 - 影响：对用户与系统：无行为变化，KV 分配结果、吞吐、内存占用均不受影响。对代码库：删除约 50 行重复逻辑，EAGLE 与 DFLASH 的 decode 分配水位计算收敛到单一宿主侧入口，降低两套 speculative 实现之间漂移的风险，为 #35223 中 "watermark 循环成为 kv_allocated_len 唯一写者 " 的长期目标铺路。对团队：后续修改页对齐或 reserve 语义时只需改动 allocation_sizing.py 一处，维护成本下降；但需要约定 reserve 的语义契约。 - 风险标记：核心分配路径重构（EAGLE/DFLASH 共享），无新增单测，依赖现有 spec 回归，reserve 语义由调用方约定，行为等价但缺少独立断言锚点

关联脉络

- PR #35265 [Spec] Page-align the DFLASH decode KV reservation: 本 PR 的直接前身：为 DFLASH decode KV 预留引入页对齐记账，本 PR 将其抽取为 EAGLE/DFLASH 共享函数，形成 " 先对齐、再共享 " 的演进时序，且两者都修改 dflash_info_v2.py。
- PR #35286 [Fix] Assert the page-aligned SWA evict floor at PD decode prealloc: 同属 tracking issue #35223 的 invariant 系列，强化页对齐不变量，与本 PR 共同服务于宿主侧页对齐记账方向。