PR #35265 完整报告

sgl-project/sglang

[Spec] Page-align the DFLASH decode KV reservation

合并时间：2026-08-19 04:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35265

执行摘要

- 一句话：DFLASH 解码 KV 预留按页对齐，修复尾部悬空记账
- 推荐动作：值得快速精读：虽然核心变更仅约 50 行，但它展示了 " 分页分配器账目一致性 " 这一容易被忽略的系统不变量，以及与 EAGLE 路径对齐的设计决策。重点阅读 prepare_for_decode 中的对齐公式和 _compute_compact_draft_seq_lens_host 的注释——后者解释了为什么 host 侧上界故意不做精确页面对齐（避免锯齿映射下冲），是同类 " host 侧无同步预估 " 场景可复用的设计范式。

功能与动机

PR body 明确说明："Round the DFLASH per-step KV reservation up to the page boundary, same as the EAGLE path (eagle_prepare_for_decode): the paged allocator hands out full pages, so an unaligned reserve strands the tail of the last page (allocated but never recorded) at page_size > 1. Part of #35223." 而 tracking issue #35223 的目标是让分配在 host 侧按整页规划，每个请求用页面对齐的 req.kv.kv_allocated_len记账，从而让free()按整页接收、彻底去掉设备读回与stream同步。本 PR 正是向 "every writer keeps kv_allocated_len page-aligned" 这一不变量迈进的具体一步。

实现拆解

1. 预留计算页面对齐（python/sglang/srt/speculative/dflash_info_v2.py）：在 DFlashDraftInputV2.prepare_for_decode 中，将原来的 reserved_len = max(cur_alloc_len, committed_len + 2 * block_size) 改为向上取整公式 nxt = max(cur, (committed_len + 2 * block_size + page_size - 1) // page_size * page_size)。page_size 来自 batch.token_to_kv_pool_allocator.page_size；当 page_size == 1 时公式退化为原值，行为不变。同步更新 num_needed_tokens += nxt - cur，使传给 alloc_for_spec_decode 的新增 token 数也按页对齐。
2. 字段语义重命名（第二个 commit "align dflash kv lens naming with eagle"）：将 reserved_seq_lens_cpu / reserved_seq_lens_sum 重命名为 nxt_kv_lens_cpu / nxt_kv_lens_sum，与 EAGLE 路径的命名对齐。filter_batch（按索引切片）与 merge_batch（batch 拼接）中的逻辑不变，只跟随改名，保持 host 侧预留数组的生命周期管理。

3. 消费方同步更新：python/sglang/srt/speculative/dflash_worker_v2.py 的 _fill_compact_seq_lens_cpu_bound 与 forward_batch_generation 中 compact / 非 compact 路径、verify 路径均改用 nxt_kv_lens_cpu / nxt_kv_lens_sum；dspark_components/dspark_draft.py 与 dspark_verify.py 同样更新。这些消费方都把该值当作 host 侧安全上界（over-estimate），向上取整不会导致越界，因为 _compute_compact_draft_seq_lens_host 使用单调包络而非精确页面对齐算术。
4. 测试配套：test/registered/unit/spec/test_dflash_overlap_hostsync.py 的 TestFilterBatchHostIndices 仅跟随字段改名，未新增针对页面对齐行为的专门断言。作者通过 /rerun-test 重跑了 test_dflash_overlap_hostsync.py 与 test/spec/dflash/test_dflash.py，均通过。

关键文件：

- python/sglang/srt/speculative/dflash_info_v2.py（模块 投机解码；类别 source；类型 core-logic；符号 DFlashDraftInputV2, prepare_for_decode, filter_batch, merge_batch）：核心变更文件：页面对齐预留计算与字段重命名都发生在这里，是 host 侧记账不变量落地的关键位置。
- python/sglang/srt/speculative/dflash_worker_v2.py（模块 投机解码；类别 source；类型 core-logic；符号 _fill_compact_seq_lens_cpu_bound, forward_batch_generation）：DFLASH worker 消费 nxt_kv_lens_cpu / nxt_kv_lens_sum 作为 host 侧安全上界，涉及 compact 与非 compact 两条规划路径，是页面对齐后语义正确性的关键验证点。
- python/sglang/srt/speculative/dspark_components/dspark_draft.py（模块 投机解码；类别 source；类型 refactor；符号 DFlashDraftInputV2）：DSpark 草稿路径同样消费 DFlashDraftInputV2 的预留字段，跟随改名保持编译与语义一致。
- python/sglang/srt/speculative/dspark_components/dspark_verify.py（模块 投机解码；类别 source；类型 refactor；符号 DFlashDraftInputV2）：DSpark verify 路径在 run_non_compact 中回退使用预留上界，跟随字段改名。
- test/registered/unit/spec/test_dflash_overlap_hostsync.py（模块 投机解码；类别 test；类型 test-coverage；符号 TestFilterBatchHostIndices, test_host_keep_list_matches_gpu_indices）：host 侧字段切片 / 拼接行为的回归测试，跟随字段改名验证 filter_batch 的 host 索引与 GPU 索引一致性。

关键符号：DFlashDraftInputV2.prepare_for_decode, DFlashDraftInputV2.filter_batch, DFlashDraftInputV2.merge_batch, DFlashDraftWorker._fill_compact_seq_lens_cpu_bound, DFlashDraftWorker.forward_batch_generation

关键源码片段

python/sglang/srt/speculative/dflash_info_v2.py

核心变更文件：页面对齐预留计算与字段重命名都发生在这里，是 host 侧记账不变量落地的关键位置。

```
# python/sglang/srt/speculative/dflash_info_v2.py
# DFlashDraftInputV2.prepare_for_decode 中的核心预留计算循环
for i, req in enumerate(batch.reqs):
```

```
    committed_len = int(req.kv_committed_len)
    # 从 req 对象读取分配水位，与 EAGLE 路径保持一致
    cur = int(req.kv.kv_allocated_len)
    # 整页记账（与 eagle_prepare_for_decode 相同）：分页分配器按整页发放，
    # 未对齐的预留会悬空最后一页的尾部——已分配但从未记录在
    # kv_allocated_len 中，后续 free 阶段会出现账目黑洞。
    nxt = max(
        cur,
        # 向上取整到页面边界：x + page_size - 1 再整除 page_size 再乘回
        (committed_len + 2 * block_size + page_size - 1)
        // page_size
        * page_size,
    )
    top_k = int(req.sampling_params.top_k)

    batch_seq_lens_cpu_t[i] = committed_len
    cur_kv_lens_cpu_t[i] = cur
    # 传给 alloc_for_spec_decode 的下一轮长度目标，必须整页对齐
    nxt_kv_lens_cpu_t[i] = nxt

    committed_seq_lens_sum += committed_len
    nxt_kv_lens_sum += nxt
    # 本次真正需要新分配的 token 数，也随对齐后的 nxt 计算
    num_needed_tokens += nxt - cur
```

python/sglang/srt/speculative/dflash_worker_v2.py

DFLASH worker 消费 nxt_kv_lens_cpu / nxt_kv_lens_sum 作为 host 侧安全上界，涉及 compact 与非 compact 两条规划路径，是页面对齐后语义正确性的关键验证点。

```
# python/sglang/srt/speculative/dflash_worker_v2.py
# host 侧上界必须是不精确页面对齐的单调包络，详见函数 docstring
def _compute_compact_draft_seq_lens_host(
    self, host_seq_lens: torch.Tensor, out: torch.Tensor
) -> None:
    """同步无关的 host 上界，用于 _compute_compact_draft_seq_lens。

    这里故意不做精确的页面对齐算术：该映射在 [window, window+page)
    区间内是非单调的锯齿形；若在过估的 host 长度（overlap 预留上界）上
    求精确值，反而可能下冲低于真实 device 值。
    min(len, window+page) 才是其单调包络，恒大于等于精确 compact len；
    消费方只需要一个安全上界，所以宁大勿小。
    """
    assert self.draft_window_size is not None
    bound = int(self.draft_window_size) + (
        self.page_size if self.page_size > 1 else 0
    )
    lens = host_seq_lens.to(dtype=torch.int64, device="cpu")
    out.copy_(torch.clamp(lens, max=bound).to(torch.int32))
```

```
def _fill_compact_seq_lens_cpu_bound(
    self,
    *,
    batch_seq_lens_cpu: Optional[torch.Tensor],
    # 本 PR 将此处字段名从 reserved_seq_lens_cpu 改为 nxt_kv_lens_cpu
    nxt_kv_lens_cpu: Optional[torch.Tensor],
    draft_prefix_lens: torch.Tensor,
    out: torch.Tensor,
) -> None:
    if batch_seq_lens_cpu is not None:
        # 优先使用已提交的 committed 前缀长度作为输入
        self._compute_compact_draft_seq_lens_host(batch_seq_lens_cpu, out=out)
    elif nxt_kv_lens_cpu is not None:
        # 页面对齐后的预留值仍是安全上界：只可能偏大，不会偏小
        self._compute_compact_draft_seq_lens_host(nxt_kv_lens_cpu, out=out)
    else:
        # 最后兜底：遗留的阻塞式 D2H 拷贝
        out.copy_(draft_prefix_lens)
```

评论区精华

本 PR 没有实质性的 review 评论或设计争论，技术论证集中在 PR body 与 tracking issue #35223 的描述中。唯一评论是作者触发的 /rerun-test test/registered/unit/spec/test_dflash_overlap_hostsync.py test/registered/spec/dflash/test_dflash.py，GitHub Actions 机器人在 1-gpu-5090 上执行 2 个测试并通过。真正的设计权衡（为什么必须页面对齐、为什么 host 上界故意不用精确页面对齐算术）以源码内注释形式保留。

- DFLASH 相关测试重跑确认 (testing): 2 个测试全部通过，无回归。

风险与影响

- 风险：
  - 行为变化面：页面对齐公式只在 page_size > 1 时改变行为；page_size == 1 时 (x + 0) // 1 * 1 恒等于 x，无任何行为差异。
  - 缓存占用小幅上升：预留向上取整后至多增加不到一页的记账量（nxt_kv_lens_sum 变大），但这是正确记账——分页分配器本来就按整页发放，页面尾部原本就无法单独分配或释放。
  - 对齐性依赖前置状态：nxt = max(cur, aligned_value) 中，若历史遗留的 cur = req.kv.kv_allocated_len 本身未对齐（本 PR 之前 DFLASH 写入的预留可能未对齐），nxt 可能仍非页面对齐。该过渡风险由配套 #35286 的断言（page-aligned SWA evict floor）兜底。
  - 测试覆盖缺口：当前测试只是字段改名跟随，没有针对 " 未对齐预留会悬空页面尾部 " 的专项断言；依赖现有 DFLASH 集成测试的回归保障。
- 影响：
  - 用户与系统：默认 page_size=1 场景无感知；在长上下文、大页分配（page_size > 1）场景下修复了 KV 记账不一致，使已分配页面尾部不再 " 游离 " 于 kv_allocated_len 之

外，为后续无设备同步的 free() 路径提供账目前提。

- 代码影响范围：涉及 DFLASH 与 DSpark 两条 spec 解码链路的 4 个源码文件 + 1 个测试文件，均为字段引用同步，无公共 API 变更。
- 团队协作：作为 #35223 host allocator 迁移的 in-flight 前置工作，统一了 DFLASH 与 EAGLE 的记账命名（nxt_kv_lens_cpu），降低后续 #35382 做逻辑 dedup 的合并成本。
- 风险标记：page_size>1 行为变化，缺少页面对齐专项断言，对齐性依赖历史 kv_allocated_len 状态

关联脉络

- PR #35286 assert the page-aligned SWA evict floor at PD decode prealloc: 同属 #35223 tracking 的不变量强化：本 PR 让 DFLASH 写入者页面对齐，而 #35286 用断言兜底 kv_allocated_len 的页面对齐性，两者相互配合。
- PR #35382 share the page-aligned decode alloc lens between EAGLE and DFLASH: 本 PR 已将 DFLASH 命名与 EAGLE 对齐（nxt_kv_lens_cpu），为 #35382 提取共享的页面对齐 decode 分配逻辑、让 watermark 循环成为 kv_allocated_len 唯一写入者做好铺垫。