PR #34234 完整报告

sgl-project/sglang

[Spec] Budget the DFLASH draft KV pool from its own attention geometry

合并时间：2026-08-10 16:28

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34234

执行摘要

- 一句话：修复 DFLASH draft KV 池按层数估算的预算偏差
- 推荐动作：建议精读。该 PR 是“显式验证假设适用范围”的典型样例：沿用的层数比例只在 draft 复用 target attention config 时才成立，新实现改为从 draft 自身几何精确计算并在失败时优雅回退。值得关注的设计：_resolve_dflash_draft_cell_size 的启动期一次性解析与 None 回退契约、_dflash_draft_cell_size 作为两个 configurator 的统一门控 accessor、以及 configure_kv_cache_dtype 的 model=None 类型放宽。测试对 0 层与 None 回退均有覆盖，后续可补充端到端显存验证。

功能与动机

PR body 指出 scale_kv_cell_size_per_token_for_dflash 按目标模型字节 / 每 token 的层数比例推导 draft 份额，其前提是“draft 与 target 共享每层 KV 尺寸（head_dim、num_kv_heads、dtype）”，这对 EAGLE/MTP 成立（draft 复用 target 的 attention config），但 “A DFLASH draft is a separately trained model, so its geometry need not match”。body 表格显示不同 KV heads 下误差显著，且 “Underestimating means pool sizing hands out more tokens than the two pools can hold”。另外 HybridSWAPoolConfigurator 按层数为 EAGLE draft 定价但无 DFLASH 项，导致 DFLASH draft 池完全落在预算之外。

实现拆解

1. 新增精确公式：python/sglang/srt/speculative/dflash_utils.py 新增 dflash_draft_cell_size_per_token()，按 KV heads 数 × (head_dim + v_head_dim) × draft 层数 × dtype 字节数 计算 draft KV 池每 token 字节数；draft_num_layers <= 0 时返回 0。
2. 启动期解析与回退：python/sglang/srt/model_executor/model_runner_components/spec_aux_hidden_state.py 在 _resolve_dflash_aux_hidden_state() 末尾调用新增的 _resolve_dflash_draft_cell_size()，结果写入 SpecAuxHiddenStateConfig.dflash_draft_cell_size_per_token；异常被捕获并 warning 后返回 None，调用方保持旧层数比例路径。
3. 统一接入两个 configurator：python/sglang/srt/model_executor/pool_configurator.py 新增门控 accessor _dflash_draft_cell_size(kvc)；DefaultPoolConfigurator 将其传给 scale_kv_cell_size_per_token_for_dflash(draft_cell_size_per_token=...) 走准确求和分支，HybridSWAPoolConfigurator 在 _compute_cell_size() 中把 self._draft_cell_size 作为 flat term 加入 all-SWA 与 hybrid 两个分支的 _cell_size。

4. KV dtype 前置解析：python/sglang/srt/mem_cache/kv_cache_dtype.py 中 configure_kv_cache_dtype() 的 model 形参从 nn.Module 放宽为 nn.Module | None，支持在 draft 模型加载前解析 KV dtype；auto 分支的 getattr(model, 'quant_config', None) 对 None 安全返回。
5. 测试配套：test/registered/unit/model_executor/test_pool_configurator.py 新增 TestDflashDraftKvBudget，覆盖公式（含 0 层）与 HybridSWA 预算随 draft term 收缩行为。

关键文件：

- python/sglang/srt/speculative/dflash_utils.py（模块 投机解码；类别 source；类型 core-logic；符号 dflash_draft_cell_size_per_token）：新增 dflash_draft_cell_size_per_token，从 draft 自身 KV 几何（heads、head_dim、v_head_dim、层数、resolved dtype）精确计算 draft KV 池的字节 /token，是整个修复的计算核心。
- python/sglang/srt/model_executor/model_runner_components/spec_aux_hidden_state.py（模块 投机配置；类别 source；类型 data-contract；符号 _resolve_dflash_draft_cell_size, SpecAuxHiddenStateConfig）：在 spec 配置解析阶段一次性解析 DFLASH draft KV 字节 /token，结果写入 SpecAuxHiddenStateConfig.dflash_draft_cell_size_per_token，失败时 warning 并返回 None 让调用方回退。
- python/sglang/srt/model_executor/pool_configurator.py（模块 内存池；类别 source；类型 data-contract；符号 _dflash_draft_cell_size）：统一门控 accessor _dflash_draft_cell_size，DefaultPoolConfigurator 传入精确值求和，HybridSWAPoolConfigurator 增加 flat term，是预算应用的核心。
- python/sglang/srt/mem_cache/kv_cache_dtype.py（模块 缓存类型；类别 source；类型 core-logic；符号 configure_kv_cache_dtype）：configure_kv_cache_dtype 的 model 形参从 nn.Module 放宽为 nn.Module | None，使 KV dtype 可以在 draft 模型加载前解析，是前置解析可行性的关键配套。
- test/registered/unit/model_executor/test_pool_configurator.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestDflashDraftKvBudget, test_bytes_per_token_from_draft_geometry, test_hybrid_swa_budget_shrinks_by_draft_pool）：新增 TestDflashDraftKvBudget 覆盖公式计算（含 0 层）与 HybridSWA 预算随 draft term 收缩的行为。

关键符号：dflash_draft_cell_size_per_token, _resolve_dflash_draft_cell_size, _dflash_draft_cell_size, configure_kv_cache_dtype

关键源码片段

python/sglang/srt/speculative/dflash_utils.py

新增 dflash_draft_cell_size_per_token，从 draft 自身 KV 几何（heads、head_dim、v_head_dim、层数、resolved dtype）精确计算 draft KV 池的字节 /token，是整个修复的计算核心。

```
# 每个 token 的 KV 字节数 = KV heads 数 × (head_dim + v_head_dim) × 层数 × dtype 字节数。
def dflash_draft_cell_size_per_token(
    *,
    draft_model_config: Any,
    draft_num_layers: int,
    draft_kv_cache_dtype: torch.dtype,
    tp_size: int,
) -> int:
    # draft 层数为 0 时不需要预留 KV 空间，直接返回 0。
    if draft_num_layers <= 0:
        return 0
    # 从 draft 自身几何读取 KV heads 与每 head 维度，不再沿用层数比例假设。
    num_kv_heads = draft_model_config.get_num_kv_heads(tp_size)
    kv_dim_per_head = draft_model_config.head_dim + draft_model_config.v_head_dim
    dtype_size = torch._utils._element_size(draft_kv_cache_dtype)
    return int(num_kv_heads * kv_dim_per_head * int(draft_num_layers) * dtype_size)
```

python/sglang/srt/model_executor/model_runner_components/spec_aux_hidden_state.py

在 spec 配置解析阶段一次性解析 DFLASH draft KV 字节 /token，结果写入 SpecAuxHiddenStateConfig.dflash_draft_cell_size_per_token，失败时 warning 并返回 None 让调用方回退。

```
# SpecAuxHiddenStateConfig 新增字段：DFLASH draft KV 的字节 /token，解析失败为 None。
# dflash_draft_cell_size_per_token: int | None = None

def _resolve_dflash_draft_cell_size(
    *,
    server_args: ServerArgs,
    draft_model_config: ModelConfig,
    draft_num_layers: int,
) -> int | None:
    # 从 sglang 运行时解析 draft worker 实际使用的 KV dtype。
    from sglang.srt.mem_cache.kv_cache_dtype import configure_kv_cache_dtype
    from sglang.srt.speculative.dflash_utils import dflash_draft_cell_size_per_token

    try:
        # 预算在 draft 模型加载前解析，因此 model 传 None，只依赖 server 参数
        # 与模型 dtype；configure_kv_cache_dtype 已放宽为接受 model=None。
        _, draft_kv_cache_dtype = configure_kv_cache_dtype(
            server_args_kv_cache_dtype=server_args.kv_cache_dtype,
            model=None,
            model_dtype=draft_model_config.dtype,
            is_draft_worker=True,
            is_dflash=True,
            speculative_draft_attention_backend=(
                server_args.speculative_draft_attention_backend
            ),
        )
```

```
            # 用 draft 自身几何精确计算，结果写入 spec 配置供 pool configurator 读取。
            return dflash_draft_cell_size_per_token(
                draft_model_config=draft_model_config,
                draft_num_layers=draft_num_layers,
                draft_kv_cache_dtype=draft_kv_cache_dtype,
                tp_size=server_args.tp_size,
            )
        except Exception as e: # noqa: BLE001
            # 解析失败时回退旧路径（层数比例），保证预算流程不中断。
            logger.warning(
                'Could not resolve DFLASH draft KV bytes/token (%s); falling back to '
                'layer-count scaling for the KV pool budget.',
                e,
            )
            return None
```

评论区精华

仓库内无技术性 review 评论。作者在 issue 评论中通过 /rerun-test 定向重跑 test_pool_configurator.py、test_basic_sanity_dflash.py、test_dflash.py、test_basic_sanity_dspark.py，github-actions bot 回报在 ubuntu-latest、1-gpu-5090、1-gpu-h100 上全部通过；随后 /tag-and-rerun-ci 打标重跑。技术决策靠 4 个 commit 自行迭代：初版实现 → 命名与 spec 约定对齐 → 完成 cell_size 重命名 → 统一共享 accessor。

- CI 定向重跑验证 DFLASH 池预算相关测试 (other): 所有 rerun 目标测试通过，PR 合并。

风险与影响

- 风险：
  1. 回退路径：_resolve_dflash_draft_cell_size 捕获所有异常并返回 None，调用方回退到旧的层数比例估算，误差问题在解析失败时依然存在，但不会中断启动。
  2. kv_cache_dtype='auto' 且 draft 启用 FP8 KV cache 时，model=None 无法读取 quant_config，会按模型 dtype（通常 bf16）解析，导致高估（池容量偏保守，不会超卖但略浪费显存）。
  3. HybridSWAPoolConfigurator._compute_cell_size 新增 flat term，会影响 DFLASH + hybrid SWA 组合的池大小；all-SWA 与 hybrid 两个分支都已覆盖，但与 _solve_pool_sizes 的联动只由单元测试保障。
  4. 变更被 is_dflash_family() 门控，EAGLE 与普通路径无行为变化。 - 影响：影响范围集中在 DFLASH/DSPARK 投机解码场景：目标 worker 的 KV 池预算从“层数比例近似”变为“按 draft 自身几何精确求和”，HybridSWAPoolConfigurator 首次把 DFLASH draft 池纳入预算，可减少容量超卖导致的 OOM 或显存浪费。对普通模型、EAGLE、draft worker 无行为变化。团队侧受益于统一 accessor，避免两个 configurator 的 DFLASH 定价逻辑漂移；新增单元测试固化了公式与预算收缩行为。 - 风险标记：池预算计算路径变更，失败回退保留旧误差，auto KV dtype 解析偏差风险，仅影响 DFLASH 家族

关联脉络

- PR #34191 [PD] Skip speculative verify scratch on prefill servers (saves num_draft_tokens x mamba pool per rank): 同属投机解码场景下的内存池预算优化，涉及 kv_cache 池容量与 speculative 内存节省。
- PR #34096 config: the KV-cache configurator reads the bags: 重构了 KV cache configurator 的配置读取路径，本 PR 的 pool_configurator 改动处于同一配置解析线上。
- PR #34189 [DSV4] Fix silent KV corruption when speculative draft tokens > 4: 同为投机解码下 KV 池正确性修复，说明投机场景内存 / 缓存预算的正确性是持续关注点。
- PR #28753 Fix/hisparse host backed max request length: 同为 pool 容量规划正确性修复，涉及 max token pool size 计算。