PR #33348 完整报告

sgl-project/sglang

[DCP] Match the replicated draft KV pool's page granularity to its allocator

合并时间：2026-08-06 02:40

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33348

执行摘要

- 一句话：DCP 下 draft KV 池页粒度对齐分配器，修复越界写
- 推荐动作：值得精读。它展示了分布式推理中 replicated pool 与 shared allocator 之间页粒度对齐的关键细节，以及如何用属性访问器把分散的条件逻辑收敛为单一事实源。关注 loc_space_scale / pool_page_size 的语义，可以避免再次引入同类 bug。

功能与动机

PR body 明确说明：Under DCP the shared allocator pages the virtual loc space in page_size * dcp_size units. #32828 sized the replicated draft KV pool for that space but left its page_size at the per-rank value, so the allocator's last page is only partly backed by pool rows and writes into its tail go out of bounds. 即分配器按 page_size * dcp_size 分页，而 pool 页太小，导致分配器最后一页的尾部写入越界，可能造成内存破坏或偶发崩溃。

实现拆解

1. 在 KVCacheConfigurator 中新增 loc_space_scale 属性：当为 draft worker 且 dcp_size > 1 时返回 dcp_size，否则返回 1，将分散在 _derive_pool_sizes 中的条件逻辑收敛为一处。
2. 新增 pool_page_size 属性，返回 get_schedule().page_size * loc_space_scale，定义 " 虚拟 loc 空间下的页粒度 "。
3. 重构 _derive_pool_sizes：用 loc_scale = self.loc_space_scale 统一缩放 max_total_num_tokens、full_max_total_num_tokens、swa_max_total_num_tokens，替换原先的 if 判断。
4. 将全部 build*_kv_pool 系列方法（DSV4、OOT DSA/MLA/MHA、Ascend SWA/MLA/MHA、DSA、MLA FP4、MLA、混合 SWA、MiniMax sparse、混合 linear、MHA FP4、MHA 等）中的 page_size=get_schedule().page_size 统一替换为 page_size=self.pool_page_size，保证每个 pool 的物理页大小与共享分配器的虚拟页对齐。
5. 无新增测试；依赖现有 CI 与 draft validation 覆盖。注意 swa_page_size 等其他参数保持独立，不受影响。

关键文件：

- python/sglang/srt/mem_cache/kv_cache_configurator.py（模块 KV 缓存；类别 source；类型 core-logic；符号 loc_space_scale, pool_page_size）：核心文件：新增 loc_space_scale 与 pool_page_size 访问器，并将所有 KV pool 构造处的 page_size 统一切换，修复 DCP 下 draft pool 越界写风险。

关键符号：loc_space_scale, pool_page_size, _derive_pool_sizes

关键源码片段

python/sglang/srt/mem_cache/kv_cache_configurator.py

核心文件：新增 loc_space_scale 与 pool_page_size 访问器，并将所有 KV pool 构造处的 page_size 统一切换，修复 DCP 下 draft pool 越界写风险。

```
# python/sglang/srt/mem_cache/kv_cache_configurator.py

# 关键设计：DCP 下 replicated draft pool 索引的是共享分配器的虚拟 loc 空间，
# 该空间按 page_size * dcp_size 分页；draft pool 必须使用同样的页粒度，
# 否则分配器最后一个页的尾部写入会越过 pool 边界。
@property
def loc_space_scale(self) -> int:
    dcp_size = self.server_args.dcp_size
    # draft worker 的 pool 是复制品而非分片，必须按整个虚拟空间缩放；
    # 非 draft worker 或单 rank 场景保持 1。
    return dcp_size if (self.is_draft_worker and dcp_size > 1) else 1

@property
def pool_page_size(self) -> int:
    # 虚拟 loc 空间下的页粒度：基础页大小乘以 loc_space_scale。
    return get_schedule().page_size * self.loc_space_scale

def _derive_pool_sizes(self, *, config: MemoryPoolConfig) -> _PoolSizes:
    # ...（省略无关部分）
    loc_scale = self.loc_space_scale
    max_total_num_tokens *= loc_scale
    if full_max_total_num_tokens is not None:
        full_max_total_num_tokens *= loc_scale
    if swa_max_total_num_tokens is not None:
        swa_max_total_num_tokens *= loc_scale
    # ...（后续逻辑不变）

# 所有 KV pool 构造统一使用 self.pool_page_size，取代原来的 get_schedule().page_size：
token_to_kv_pool = pool_cls(
    max_total_num_tokens,
    page_size=self.pool_page_size, # 与共享分配器的虚拟页粒度保持一致
    dtype=self.kv_cache_dtype,
    # ...
)
```

评论区精华

Review 流程极简：hnyls2002 直接 APPROVED，无公开评论。作者曾通过 /rerun-failed-ci 重跑过一次 CI（extra 任务曾失败，最终通过）。没有出现设计或正确性争议。

- 暂无高价值评论线程

风险与影响

- 风险：影响面限定在 draft worker 且 dcp_size > 1 的 DCP 部署：pool_page_size 变为原 page_size 的 dcp_size 倍，每页内存增大，页表项减少，尾部对齐可能产生少量内存浪费，但总容量不变。所有 pool 构造路径（包括 NPU 的 Ascend 路径）都已切换，需确认 NPU 后端对 page_size 的假设是否兼容；目前没有专门针对该组合的单元测试，回归依赖 e2e 测试。另外，若未来共享分配器的分页逻辑再次变化，需同步维护 pool_page_size 访问器。
- 影响：对使用 DCP 多 rank + draft worker（如 DeepSeek 系列）的用户，修复了潜在的非确定性越界写入，避免偶发的内存腐蚀、崩溃或错误结果。对单 rank 或非 draft worker 完全无行为变化。对团队来说，这是一个小而重要的正确性修复，代码可读性也因访问器收敛而提升。
- 风险标记：核心内存池路径变更，缺少直接测试覆盖

关联脉络

- PR #32828 [DCP] Size the replicated draft KV pool for the DCP virtual loc space: 本 PR 是 #32828 的 follow-up。#32828 将 draft pool 容量按 dcp_size 缩放但未调整 page_size，导致本 PR 修复的页粒度不匹配问题。