PR #32477 完整报告

sgl-project/sglang

[Kernel] Skip KV writes to reserved padding slots

合并时间：2026-07-29 09:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32477

执行摘要

- 一句话：跳过保留 KV padding 槽位的无效写入
- 推荐动作：值得精读，尤其是理解 NaN 污染机制和 JIT 内核参数扩展方式。设计决策简洁有效，测试覆盖全面。

功能与动机

CUDA-graph padding rows 可能包含未定义的 K/V 值，但仍指向保留的 padding slot（索引 0）。在 sliding-window attention 中，page-table padding 也指向该保留页。部分有效的 tile 可能在屏蔽无效行之前加载该页，导致 NaN 通过 0 * NaN 污染结果。

实现拆解

1. 新增 reserved_skip_index 参数：在 kvcache.cuh 的 StoreKVCacheParams 结构体中添加 reserved_skip_index 字段，并在内核循环中增加条件判断 if (index != reserved_skip_index) 以跳过写入。
2. 修改 Python 接口：在 kvcache.py 的 store_cache 函数中增加 reserved_skip_index 参数，默认值为 0（保留槽索引），-1 表示禁用跳过。
3. 更新现有测试：将 test_store_cache、test_store_cache_dtypes、test_store_cache_int32_indices 中的随机索引生成改为偏移 +1，避免使用索引 0。
4. 新增针对性测试：添加 test_store_cache_reserved_skip_index 测试保留槽跳过，以及 test_store_cache_zero_index_can_be_written_when_skip_disabled 测试禁用跳过后索引 0 可正常写入。

关键文件：

- python/sglang/kernels/jit/csrc/elementwise/kvcache.cuh（模块 JIT 内核；类别 source；类型 core-logic；符号 StoreKVCacheParams, store_kvcache, StoreKVCacheKernel）：核心修改：在内核参数结构体中添加 reserved_skip_index，并在写入前增加条件判断跳过保留槽。
- python/sglang/kernels/ops/kvcache/kvcache.py（模块 KV 缓存；类别 source；类型 infrastructure；符号 store_cache）：Python 接口层：为 store_cache 函数新增 reserved_skip_index 参数，传递到 C++ 内核。
- test/registered/kernels/ops/kvcache/test_store_cache.py（模块 测试；类别 test；类型 test-coverage；符号 test_store_cache_reserved_skip_index,

test_store_cache_zero_index_can_be_written_when_skip_disabled）：测试文件：修改现有测试避免使用索引 0，新增两个测试函数覆盖保留槽跳过逻辑和 opt-out 路径。

关键符号：store_cache, store_kvcache, StoreKVCacheKernel::Instantiate

关键源码片段

python/sglang/kernels/jit/csrc/elementwise/kvcache.cuh

核心修改：在内核参数结构体中添加 reserved_skip_index，并在写入前增加条件判断跳过保留槽。

```
// 在 StoreKVCacheParams 结构体中增加保留槽索引字段
struct StoreKVCacheParams {
  // ... 其他字段
  int64_t reserved_skip_index; // 如果 index == reserved_skip_index，则跳过该位置的 KV 写入
};

// 内核函数：存储 KV 到 cache，跳过保留槽
__global__ void store_kvcache(const __grid_constant__ StoreKVCacheParams params) {
  const auto& [k_input, v_input, k_cache, v_cache, indices,
               stride_k, stride_v, stride_cache, stride_indices, batch_size,
               size_limit, reserved_skip_index] = params; // 新增保留槽索引
  if (item_id >= batch_size) return;
  // ... 索引计算 ...
  // 关键修改：跳过对保留槽的写入
  if (index != reserved_skip_index) {
    copy_kv_warp<kSplitSize>(k_src, v_src, k_dst, v_dst);
  }
  PDLTriggerSecondary<kUsePDL>();
}

// 内核启动函数：新增 reserved_skip_index 参数并传入参数字段
struct StoreKVCacheKernel {
  // ...
  static void Instantiate(... , const int64_t reserved_skip_index) {
    // ...
    StoreKVCacheParams params = {
      // ...
      .reserved_skip_index = reserved_skip_index,
    };
    // ...
  }
};
```

python/sglang/kernels/ops/kvcache/kvcache.py

Python 接口层：为 store_cache 函数新增 reserved_skip_index 参数，传递到 C++ 内核。

```
def store_cache(
    k: torch.Tensor,
```

```
    v: torch.Tensor,
    k_cache: torch.Tensor,
    v_cache: torch.Tensor,
    indices: torch.Tensor,
    row_bytes: int = 0,
    num_split: int = 0,
    size_limit: int = 0,
    reserved_skip_index: int = 0, # 新增参数，默认跳过索引 0（CUDA-graph padding 保留槽）
) -> None:
    """
    Store key and value tensors into KV cache at specified indices.
    ...
    reserved_skip_index (int): If nonnegative, writes targeting this index
        are skipped. Defaults to the reserved CUDA-graph padding slot 0;
        pass -1 to disable skipping.
    """
    # ...
    # 将 reserved_skip_index 传入内核
    module(
        k, v, k_cache, v_cache,
        indices,
        num_split,
        size_limit,
        reserved_skip_index, # 透传到 JIT 内核
    )
```

test/registered/kernels/ops/kvcache/test_store_cache.py

测试文件：修改现有测试避免使用索引 0，新增两个测试函数覆盖保留槽跳过逻辑和 opt-out 路径。

```
# 新增测试：验证保留槽被跳过 @pytest.mark.parametrize("index_dtype", [torch.int32, torch.int64]) @pytest.mark.parametrize("num_split", [1, 2, 4]) deftest_store_cache_reserved_skip_index(     index_dtype: torch.dtype, num_split: int ) -> None:     element_dim = 1024     k = torch.randn((4, element_dim), dtype=DTYPE, device=DEVICE)     v = torch.randn((4, element_dim), dtype=DTYPE, device=DEVICE)  # 模拟 CUDA-graph padding 行包含 NaN     k[[0, 2]] = torch.nan     v[[0, 2]] = torch.nan     k_cache = torch.randn((SMALL_CACHE, element_dim), dtype=DTYPE, device=DEVICE)     v_cache = torch.randn((SMALL_CACHE, element_dim), dtype=DTYPE, device=DEVICE)     reserved_k_before = k_cache[0].clone() reserved_v_before = v_cache[0].clone()     indices = torch.tensor([0, 7, 0, 9], dtype=index_dtype, device=DEVICE) # 包含重复的 0     store_cache(         k,         v,         k_cache,         v_cache,         indices,         num_split=num_split,     )     # 验证：保留槽索引 0 不应被修改（因为 default reserved_skip_index=0) torch.testing.assert_close(k_cache[0], reserved_k_before, rtol=0.0, atol=0.0) torch.testing.assert_close(v_cache[0], reserved_v_before, rtol=0.0, atol=0.0)     # 验证：非保留槽被正确写入     torch.testing.assert_close(k_cache[indices[1].long()], k[1], rtol=0.0, atol=0.0)     torch.testing.assert_close(v_cache[indices[1].long()], v[1],
```

```
rtol=0.0, atol=0.0) # 新增测试：验证 opt-out 路径（reserved_skip_index=-1）下索引 0 可正常写入 deftest_store_cache_zero_index_can_be_written_when_skip_disabled() -> None:    # ... 设置数据，indices 包含 0 ...    store_cache(k, v, k_cache, v_cache, indices, reserved_skip_index=-1)    # 验证索引 0 被正确写入 torch.testing.assert_close(k_cache[0], k[0], ...) torch.testing.assert_close(v_cache[0], v[0], ...)
```

评论区精华

本 PR 无实质性 review 讨论。两位 reviewer 均直接批准，仅机器人评论已过时。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。变更集中在 JIT 内核的参数传递和条件判断，不影响已存在的正确路径。新增参数默认行为与之前一致（跳过索引 0），opt-out 通过传递 -1 实现。测试覆盖了 int32/int64 索引、不同 num_split 值、以及禁用跳过路径。
- 影响：影响范围限定在 JIT KV-cache store 内核，特别是使用 CUDA-graph 的 sliding-window attention 场景。修复可避免罕见的 NaN 污染问题，提升数值稳定性。调用方无需修改，除非希望自定义保留槽索引。
- 风险标记：核心路径变更，测试覆盖全面

关联脉络

- 暂无明显关联 PR