

PR #48411 完整报告

vllm-project/vllm

[Bugfix] Include inline per-token-head scales in offloaded page transfer width

合并时间: 2026-07-14 21:07

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48411>

执行摘要

- 一句话: 修复 KV 缓存卸载时 per-token-head 量化数据截断
- 推荐动作: 建议精读。此 PR 修复了一个难以发现的静默数据损坏 bug, 展示了对量化和内存布局细节的深刻理解。重构方式 (内联缩放到独立属性) 提供了良好的代码复用性和可测试性, 值得参考。

功能与动机

Per-token-head 量化的 KV 页面在每个 (head, token) 单元内嵌入了 fp32 缩放因子, 这些数据位于 `real_page_size_bytes` 预算之外。卸载连接器使用 `real_page_size_bytes` 作为块传输宽度, 导致尾部 head/token 数据及缩放因子被丢弃, 恢复时带来损坏块。此问题影响所有 TP 配置, 包括单实例恢复。

实现拆解

1. 重构 `AttentionSpec` 的页面大小计算 (`vllm/v1/kv_cache_interface.py`) :
 - 新增 `unpadded_page_size_bytes` 属性, 返回 `real_page_size_bytes` 加上 per-token-head 缩放因子的大小。原先这部分逻辑内联在 `page_size_bytes` 中。
 - 修改 `page_size_bytes`, 在无 padding 时直接委托给 `unpadded_page_size_bytes`; 有 `page_size_padded` 时沿用原有逻辑。
2. 更新卸载连接器 (`vllm/distributed/kv_transfer/kv_connector/v1/offloading/worker.py`) :
 - 将 `unpadded_page_size_bytes` 的赋值从 `layer_kv_cache_spec.real_page_size_bytes` 改为 `layer_kv_cache_spec.unpadded_page_size_bytes`, 确保传输宽度包含内嵌缩放因子。
3. 新增单元测试 (`tests/v1/core/test_kv_cache_utils.py`) :
 - `test_unpadded_page_size_without_quant_matches_real_page`: 验证无量化时 `unpadded_page_size_bytes` 等于 `real_page_size_bytes`。
 - `test_unpadded_page_size_includes_per_token_head_scales`: 验证 per-token-head 量化时 `unpadded_page_size_bytes` 正确包含缩放因子。
 - `test_page_size_padded_wins`: 验证 `page_size_padded` 生效时优先级正确。
 - 为支持测试, `new_kv_cache_spec` 新增 `kv_quant_mode` 参数。

关键文件：

- vllm/v1/kv_cache_interface.py (模块 KV 缓存接口；类别 source；类型 core-logic；符号 `page_size_bytes`, `unpadded_page_size_bytes`)：新增 `unpadded_page_size_bytes` 属性，将 per-token-head 缩放因子计算从 `page_size_bytes` 中提取出来；`page_size_bytes` 改为委托实现。
- vllm/distributed/kv_transfer/kv_connector/v1/offloading/worker.py (模块 卸载连接器；类别 source；类型 core-logic)：将 `unpadded_page_size_bytes` 的赋值从 `real_page_size_bytes` 改为 `unpadded_page_size_bytes`，使传输宽度包含内嵌缩放因子。
- tests/v1/core/test_kv_cache_utils.py (模块 KV 缓存工具；类别 test；类型 test-coverage；符号 `test_unpadded_page_size_without_quant_matches_real_page`, `test_unpadded_page_size_includes_per_token_head_scales`, `test_page_size_padded_wins`)：新增三个测试用例，覆盖无量化、per-token-head 量化、padding 优先级场景；导入 `KVQuantMode` 并扩展 `new_kv_cache_spec` 工厂函数。

关键符号： `unpadded_page_size_bytes`, `page_size_bytes`

关键源码片段

vllm/v1/kv_cache_interface.py

新增 `unpadded_page_size_bytes` 属性，将 per-token-head 缩放因子计算从 `page_size_bytes` 中提取出来；`page_size_bytes` 改为委托实现。

```
# vllm/v1/kv_cache_interface.py
```

```
@dataclass(frozen=True, kw_only=True)
```

```
class AttentionSpec(KVCacheSpec):
```

```
    num_kv_heads: int
```

```
    head_size: int
```

```
    dtype: torch.dtype
```

```
    kv_quant_mode: KVQuantMode = KVQuantMode.NONE
```

```
    page_size_padded: int | None = None
```

```
    indexes_kv_by_block_stride: bool = False
```

```
@property
```

```
def unpadded_page_size_bytes(self) -> int:
```

```
    # 返回未 padding 的页大小，包含 per-token-head 缩放因子
```

```
    unpadded = self.real_page_size_bytes
```

```
    if self.kv_quant_mode.is_per_token_head:
```

```
        # 每个 (head, token) 单元包含 2 个 fp32 缩放因子 (key 和 value)
```

```
        unpadded += (
```

```
            2 * self.block_size * self.num_kv_heads * get_dtype_size(torch.float32)
```

```
        )
```

```
    return unpadded
```

```
@property
```

```
def page_size_bytes(self) -> int:
```

```
    # 若明确指定了 padding 大小，则返回 padding 值
```

```

    if self.page_size_padded is not None:
        assert self.page_size_padded >= self.unpadded_page_size_bytes
        return self.page_size_padded
    # 否则返回未 padding 的大小
    return self.unpadded_page_size_bytes

@property
def real_page_size_bytes(self) -> int:
    # 仅含 KV 数据（不含缩放因子）的裸页大小
    ... # 计算逻辑不变

```

tests/v1/core/test_kv_cache_utils.py

新增三个测试用例，覆盖无量化、per-token-head 量化、padding 优先级场景；导入 `KVQuantMode` 并扩展 `new_kv_cache_spec` 工厂函数。

```

# tests/v1/core/test_kv_cache_utils.py

def test_unpadded_page_size_without_quant_matches_real_page():
    # 无量化时，unpadded_page_size_bytes 应与 real_page_size_bytes 一致
    spec = new_kv_cache_spec()
    assert spec.unpadded_page_size_bytes == spec.real_page_size_bytes
    assert spec.page_size_bytes == spec.unpadded_page_size_bytes

def test_unpadded_page_size_includes_per_token_head_scales():
    # Per-token-head 量化时，缩放因子应计入 unpadded_page_size_bytes
    spec = new_kv_cache_spec(
        dtype=torch.uint8, kv_quant_mode=KVQuantMode.FP8_PER_TOKEN_HEAD
    )
    scales = 2 * spec.block_size * spec.num_kv_heads * 4 # 每个 head 2 个 fp32
    assert spec.unpadded_page_size_bytes == spec.real_page_size_bytes + scales
    assert spec.page_size_bytes == spec.unpadded_page_size_bytes

def test_page_size_padded_wins():
    # 显式指定 page_size_padded 时，page_size_bytes 应返回该值
    spec = new_kv_cache_spec(
        dtype=torch.uint8,
        kv_quant_mode=KVQuantMode.FP8_PER_TOKEN_HEAD,
        page_size_padded=65536,
    )
    assert spec.page_size_bytes == 65536

```

评论区精华

- orozery 建议将测试文件从 `tests/v1/kv_offload/test_kv_cache_utils.py` 移至已有的 `tests/v1/core/test_kv_cache_utils.py`，以复用 `new_kv_cache_spec` 工具函数。作者已采纳并完成迁移。

- orozery 提议将属性名从 `content_size_bytes` 改为 `unpadded_page_size_bytes`，因其语义更明确（指代未 padding 的页面大小），与 `page_size_bytes` 形成对应。作者已采纳，并在后续提交中完成重命名。
- orozery 最终批准了该 PR。
- 测试文件位置 (testing): 作者将测试代码迁移至 `tests/v1/core/test_kv_cache_utils.py`。
- 属性命名: `content_size_bytes` vs `unpadded_page_size_bytes` (design): 作者采纳建议，重命名为 `unpadded_page_size_bytes`。

风险与影响

- 风险：变更范围小且经过专门测试覆盖，风险较低。主要风险在于：
 - 第三方代码如果直接依赖 `page_size_bytes` 属性中的 per-token-head 缩放因子内联逻辑，可能因重构而出现行为变化。但 `page_size_bytes` 的返回结果与重构前一致（内部逻辑从内联变为委托），因此兼容性风险低。
 - 单组模型目前走跨层卸载路径，与 per-token-head 量化存在独立的兼容性问题（分配时缺少缩放因子打包），PR 作者已注明将单独报告，不影响本更改。
- 影响：
 - 用户影响：使用 per-token-head 量化 KV 缓存（如 FP8_PER_TOKEN_HEAD）且启用了 CPU 卸载功能的用户将获得正确的数据恢复，消除静默数据损坏。
 - 系统影响：卸载传输宽度增加（每个 block 多出 $2 * \text{block_size} * \text{num_kv_heads} * 4$ 字节），但这是正确性所必需的。
 - 开发影响：AttentionSpec 新增了清晰的 `unpadded_page_size_bytes` 属性，便于其他卸载或传输逻辑复用，也使 `page_size_bytes` 的职责更单一。
- 风险标记：单组模型跨层卸载路径尚存不兼容问题

关联脉络

- PR #47754 [Test] Enable KV cache events for HMA models in CPU offloading test: 同为 KV 缓存卸载相关 PR，涉及同一测试文件 `tests/v1/kv_connector/unit/test_offloading_connector.py` 及卸载连接器模块。