PR #50194 完整报告

vllm-project/vllm

[CPU] Fix FP8 attention scratchpad sizing

合并时间：2026-07-29 15:34

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50194

执行摘要

- 一句话：修复 CPU FP8 注意力调度器临时缓冲区大小计算错误
- 推荐动作：该 PR 值得所有使用 CPU 后端的用户关注，尤其是启用 FP8 量化的场景。其核心设计决策——将 KV cache dtype 信息传递至调度器，确保 tile 大小计算与运行时一致——是值得借鉴的 bug 修复模式。建议阅读 csrc/cpu/cpu_attn.cpp 中的改动以理解调度器大小计算逻辑。

功能与动机

CPU 注意力调度器在计算 FP8 KV tile 几何尺寸时，使用了 sizeof(kv_cache_t)，而该类型在 BF16 查询和单字节 FP8 KV cache 条目组合下，会选出比 BF16 支持的临时缓冲区更大的 tile，导致 AMX 大预填充期间越界写入，输出结果损坏。PR body 中测试结果显示，修复前 gsm8k 准确率仅 0.67，且存在 27 次重复 '!' 符号的 corruption；修复后准确率提升至 0.87，corruption 消失。

实现拆解

1. C++ 核心函数 get_scheduler_metadata 新增 kv_cache_dtype 参数（csrc/cpu/cpu_attn.cpp）：函数签名增加 const std::string& kv_cache_dtype，并使用 parse_fp8_kv_dtype 将其转换为枚举索引 kv_cache_idx，传递给 CPU_ATTN_DISPATCH 宏，使得在调度器内部能够根据实际 KV cache 类型（如 fp8_e4m3）正确获取 sizeof(kv_cache_t)，而非始终假定为 BF16 大小。
2. Torch 绑定更新（csrc/cpu/torch_bindings.cpp）：在声明和注册 get_scheduler_metadata 的地方同步添加 str kv_cache_dtype="auto" 参数，保持向前兼容。
3. Python 接口层传递参数（vllm/v1/attention/backends/cpu_attn.py、vllm/_custom_ops.py）：CPUAttentionMetadataBuilder 将配置的 kv_cache_dtype 存储为 self.kv_cache_dtype，并在调用 ops.cpu_attn_get_scheduler_metadata 时传入；vllm/_custom_ops.py 中 cpu_attn_get_scheduler_metadata 函数新增 kv_cache_dtype 默认值 "auto"，并转发给底层 C++ 函数。
4. 新增回归测试（tests/kernels/attention/test_cpu_attn.py）：新增 test_varlen_with_paged_kv_fp8_large_prefill_amx 测试函数，使用大块大小（block_size=2176）、多个长序列（每个 1024 tokens）和 FP8 KV cache 在 AMX ISA 下执行注意力计算，验证不会出现越界写入。同时将 varlen_with_paged_kv 辅助函数中对

cpu_attn_get_scheduler_metadata 的调用也传入 kv_cache_dtype 参数，确保现有测试正确传递该参数。

关键文件：

- csrc/cpu/cpu_attn.cpp（模块 CPU 内核；类别 source；类型 core-logic；符号 get_scheduler_metadata）：核心修复文件：get_scheduler_metadata 函数新增 kv_cache_dtype 参数，并利用 parse_fp8_kv_dtype 获取正确的 KV cache 类型索引，传递给 CPU_ATTN_DISPATCH，使调度器能够基于实际 KV cache 类型计算 tile 大小，而非始终假定与 Query 相同 dtype。
- tests/kernels/attention/test_cpu_attn.py（模块 CPU 注意力；类别 test；类型 test-coverage；符号 test_varlen_with_paged_kv_fp8_large_prefill_amx, varlen_with_paged_kv）：新增回归测试 test_varlen_with_paged_kv_fp8_large_prefill_amx，使用大块大小（2176）和长序列（4 个 1024 token）在 AMX+FP8 条件下验证越界写入问题是否修复；同时修改辅助函数 varlen_with_paged_kv 使其在调用 cpu_attn_get_scheduler_metadata 时传递 kv_cache_dtype 参数。
- vllm/v1/attention/backends/cpu_attn.py（模块 V1 引擎；类别 source；类型 core-logic；符号 CPUAttentionMetadataBuilder.init, CPUAttentionMetadataBuilder.build）：Python 侧主调用入口：CPUAttentionMetadataBuilder.__init__ 保存 self.kv_cache_dtype，并在 build 方法中将其传递给 ops.cpu_attn_get_scheduler_metadata，从而将用户配置的 KV cache dtype 传递到 C++ 内核。
- csrc/cpu/torch_bindings.cpp（模块 CPU 内核；类别 source；类型 core-logic；符号 get_scheduler_metadata）：C++ 侧函数声明和 Torch op 注册：同步更新 get_scheduler_metadata 的声明和 TORCH_LIBRARY 定义，新增 str kv_cache_dtype="auto" 参数，确保 Python 侧能够正确调用。
- vllm/_custom_ops.py（模块 自定义操作；类别 source；类型 core-logic；符号 cpu_attn_get_scheduler_metadata）：Python 自定义操作封装层：cpu_attn_get_scheduler_metadata 函数新增 kv_cache_dtype 参数并透传给底层 C++ 操作，确保参数正确传递。

关键符号：get_scheduler_metadata, CPUAttentionMetadataBuilder.build, cpu_attn_get_scheduler_metadata, test_varlen_with_paged_kv_fp8_large_prefill_amx, varlen_with_paged_kv

关键源码片段

csrc/cpu/cpu_attn.cpp

核心修复文件：get_scheduler_metadata 函数新增 kv_cache_dtype 参数，并利用 parse_fp8_kv_dtype 获取正确的 KV cache 类型索引，传递给 CPU_ATTN_DISPATCH，使调度器能够基于实际 KV cache 类型计算 tile 大小，而非始终假定与 Query 相同 dtype。

```
// csrc/cpu/cpu_attn.cpp: get_scheduler_metadata
// 新增 kv_cache_dtype 参数，用于在 FP8 KV cache 场景下正确计算 tile 大小
// 之前版本始终假定 KV cache 与 Query 具有相同 dtype（如 BF16），
// 导致 FP8 时选择的 tile 过大，超出 scratchpad 容量，造成越界写入。
```

```
torch::Tensor get_scheduler_metadata(
    const int64_t num_req, const int64_t num_heads_q,
    const int64_t num_heads_kv, const int64_t head_dim,
    const torch::Tensor& seq_lens, at::ScalarType dtype,
    const torch::Tensor& query_start_loc, const bool causal,
    const int64_t window_size, const std::string& isa_hint,
    const bool enable_kv_split,
    const std::optional<torch::Tensor>& dynamic_causal,
    const std::string& kv_cache_dtype) { /* <-- 新增参数 */
  // ... ISA 解析代码省略 ...

  cpu_attention::AttentionScheduler::ScheduleInput input;
  // ... 填充 input 字段 ...

  // 将 kv_cache_dtype 字符串转换为枚举索引，传递给调度器
  // 0 = kAuto, 1 = kFp8E4M3, 2 = kFp8E5M2
  const int64_t kv_cache_idx =
      static_cast<int64_t>(parse_fp8_kv_dtype(kv_cache_dtype));

  VLLM_DISPATCH_FLOATING_TYPES(dtype, "get_scheduler_metadata", [&]() {
    // 传入 kv_cache_idx 以区分不同 KV cache 类型
    CPU_ATTN_DISPATCH(head_dim, isa, kv_cache_idx, [&]() {
      // 关键修复：使用 sizeof(attn_impl::kv_cache_t) 替代 sizeof(scalar_t)
      // 之前使用 sizeof(scalar_t) 始终为 BF16 大小（2 字节），
      // 而 FP8 时 kv_cache_t 仅 1 字节，导致 tile 选择过大。
      input.elem_size = sizeof(attn_impl::kv_cache_t);
      // ... 其余缓冲区大小保持不变 ...
    });
  });

  cpu_attention::AttentionScheduler scheduler;
  torch::Tensor metadata = scheduler.schedule(input);
  return metadata;
}
```

tests/kernels/attention/test_cpu_attn.py

新增回归测试 test_varlen_with_paged_kv_fp8_large_prefill_amx，使用大块大小（2176）和长序列（4 个 1024 token）在 AMX+FP8 条件下验证越界写入问题是否修复；同时修改辅助函数 varlen_with_paged_kv 使其在调用 cpu_attn_get_scheduler_metadata 时传递 kv_cache_dtype 参数。

```
# tests/kernels/attention/test_cpu_attn.py
# 新增的回归测试：模拟大预填充 + FP8 KV cache + AMX ISA 场景
# 用于验证 tile 大小计算是否正确，是否会越界写入
@pytest.mark.skipif(not torch.cpu._is_amx_tile_supported(),
                    reason="no AMX support.")
def test_varlen_with_paged_kv_fp8_large_prefill_amx() -> None:
    # 使用 4 个长序列（每个 1024 tokens），大 block_size 以触发大 tile 计算
    # head_size=256，num_heads=16/2，使 AMX tile 调度器产生大 tile
```

```
    # kv_cache_dtype="fp8_e4m3" 令 KV cache 元素大小为 1 字节
    varlen_with_paged_kv(
        seq_lens=[(1024, 1024)] * 4,
        num_heads=(16, 2),
        head_size=256,
        sliding_window=None,
        dtype=torch.bfloat16,
        block_size=2176, # 大块大小，增加 tile 使用压力
        soft_cap=None,
        num_blocks=4,
        use_alibi=False,
        use_sink=False,
        isa="amx",
        kv_cache_dtype="fp8_e4m3", # FP8 类型，之前会导致 tile 过大
    )
```

评论区精华

review 讨论集中在如何修复调度器大小计算与运行时不一致的问题。bigPYJ1151 在 csrc/cpu/cpu_attn_impl.hpp 第 1553 行评论指出，更好的解决方案是让 get_scheduler_metadata 感知 FP8 KV cache 情况，使 schedule 中的 sizing 与 runtime 保持一致，当前代码总是假定 KV cache 与 Query 具有相同 dtype。tianmu-li 回复同意该方案，并已按照建议更新代码，使得 get_scheduler_metadata 接收 kv_cache_dtype 参数，从而调度器 sizing 使用与运行时 tile 计算相同的 KV cache 类型。

- 让 get_scheduler_metadata 感知 FP8 KV cache 类型 (design): tianmu-li 采纳建议，在 get_scheduler_metadata 中添加 kv_cache_dtype 参数，并利用 parse_fp8_kv_dtype 解析类型，确保 tile 大小计算与运行时使用的 KV cache 类型一致。

风险与影响

- 风险：本 PR 修改了 CPU 注意力调度器的核心路径，涉及 C++ 内核和 Python 接口。主要风险包括： 1) 新增参数 kv_cache_dtype 默认值为 "auto"，对非 FP8 路径保持向后兼容，回归风险较低； 2) 测试覆盖了 AMX+FP8 大预填充场景，但未覆盖其他 ISA（如 VEC/VEC16）下 FP8 路径，可能存在未被发现的类似问题； 3) 修改了 C++ 函数签名和 Python-Op 绑定，若与其他未同步的分支合并可能引入编译错误。
  - 影响：影响范围限于使用 CPU 后端且启用了 FP8 KV cache 的用户，特别是使用 AMX 指令集的场景。对于非 FP8 用户无影响。修复后，FP8 KV cache 下的准确率从 0.67 提升至 0.87，消除了输出中的重复 '!' corruption 符号，属于正确性关键修复。
  - 风险标记：核心路径变更，C++/Python 接口修改

关联脉络

- 暂无明显关联 PR