PR #27380 完整报告

sgl-project/sglang

[AMD] Add unified kv attention support in dpsk-v4

合并时间：2026-06-10 14:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27380

执行摘要

- 一句话：为 AMD DeepSeek-V4 添加 unified KV attention 后端
- 推荐动作：该 PR 展示了如何在现有复杂代码库中通过 gate 引入重大新功能，值得关注其设计权衡（统一池 vs 独立池、CG safety）。对于 AMD 推理栈开发者，建议仔细 review 内核健壮性并补充 AMD 硬件上的测试。

功能与动机

PR 作者指出，移植 ATOM 的稀疏注意力内核可在 AMD 上获得显著性能提升（PR body: "porting ATOM's sparse attention kernels, which has great perf."）。现有的三池布局需要每次 forward 动态组合 KV 扁平张量，阻碍 CUDA Graph 捕获，统一内存页面池使索引变得静态。

实现拆解

实现分为以下步骤：

1. 内存池改造：在 deepseek_v4_memory_pool.py 中新增 DeepSeekV4UnifiedKVPool 类，为每层分配一个 [swa_pages + compress_pages, head_dim] 的 bf16 统一页面池。替代原有的 DeepSeekV4SingleKVPool 三池独立布局。
2. 存储路径适配：修改 fused_qk_norm_rope_store.py 和 jit_kernel/dsv4/compress.py，为 fused_norm_rope 内核添加 bf16_store 参数，使其能够将 SWA KV 和压缩 KV 直接写入 bf16 统一池。
3. 注意力内核移植：在新增的 unified_kv_kernels/ 目录下创建四个文件：
   - paged_decode.py：核心 decode 注意力内核，使用 online-softmax + KV split/fused 两条路径。
   - paged_prefill.py：prefill 注意力内核，同时从统一池和当前 token 的 flat KV 读取。
   - paged_decode_indices.py：用于构建 per-token 索引的 Triton 核，替代之前的 CPU 构建 + index_copy_。
   - runtime.py：胶水模块，包含 SWA 散射（store_swa_into_unified）、compress tail 填充、索引流构建函数（build_decode_streams、build_prefill_indices）以及统一的 decode / prefill 分发函数。
4. 后端集成：在 deepseek_v4_backend_hip_radix.py 中新增 _attach_unified_kv_decode_streams 和 _attach_unified_kv_prefill_meta 方法，在每次

forward 时根据 metadata 构建索引流，并调用统一注意力内核。同时对 DSV4AttnMetadata 增加了大量 unified kv 相关字段。

5. 环境门控：新增 env_gate.py，通过 is_unified_kv_triton() 检查环境变量 SGLANG_HACK_FLASHMLA_BACKEND 是否为 unified_kv_triton，从而控制是否启用新路径。默认禁用，保证不影响现有 NVIDIA 和 AMD 流程。

关键文件：

- python/sglang/srt/layers/attention/dsv4/unified_kv_kernels/paged_decode.py（模块 注意力核；类别 source；类型 core-logic；符号 _cu_count, _kernel_config, _prev_pow2, _kv_splits_heuristic）：核心解码注意力内核，包含 split-K 和 fused 两条路径，实现 online-softmax 和 CUDAGraph 安全设计。该文件是 unified KV 后端的计算核心。
- python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 DeepSeekV4UnifiedKVPool, init, get_unified_kv, get_buf_infos）：新增 DeepSeekV4UnifiedKVPool 类，定义了 unified KV 池的内存布局，是其他所有 kernel 的基础数据结构。同时对 DeepSeekV4TokenToKVPool 添加了 sliding_window 参数并门控启用 unified pool。
- python/sglang/srt/layers/attention/dsv4/unified_kv_kernels/runtime.py（模块 运行时；类别 source；类型 core-logic；符号 _swa_scatter_kernel, store_swa_into_unified, _lengths_to_indptr, decode）：提供 SWA 散射、compress tail 填充、索引流构建等关键胶水函数，是连接原始 metadata 结构与注意力内核的桥梁。

关键符号：_sparse_attn_v4_paged_decode_triton, _sparse_attn_v4_paged_prefill_kernel, store_swa_into_unified, build_decode_streams, build_prefill_indices, _attach_unified_kv_decode_streams, fill_compress_tail, write_v4_paged_decode_indices, is_unified_kv_triton

关键源码片段

python/sglang/srt/layers/attention/dsv4/unified_kv_kernels/runtime.py

提供 SWA 散射、compress tail 填充、索引流构建等关键胶水函数，是连接原始 metadata 结构与注意力内核的桥梁。

```
# SWA ring scatter kernel (Triton JIT)
@triton.jit
def _swa_scatter_kernel(
    kv_ptr, # [T, D] bf16 input KV
    state_slot_ptr, # [T] int request slot per token
    positions_ptr, # [T] int token position
    final_pos_ptr, # [T] int request final position (for sliding window boundary check)
    unified_ptr, # [pages, D] bf16 target unified pool
    n_rows,
    ring_stride, # SWA ring stride per slot
    win: tl.constexpr,
    D: tl.constexpr,
    HAS_FINAL: tl.constexpr,
    BLOCK_D: tl.constexpr,
```

```
):
    """Scatter KV into the correct ring location of the unified pool."""
    row = tl.program_id(0)
    if row >= n_rows:
        return
    pos = tl.load(positions_ptr + row)
    if HAS_FINAL:
        fp = tl.load(final_pos_ptr + row)
        if pos <= fp - win: # outside window, skip
            return
    s = tl.load(state_slot_ptr + row)
    loc = s * ring_stride + (pos % ring_stride) # ring index formula
    offs = tl.arange(0, BLOCK_D)
    mask = offs < D
    vals = tl.load(kv_ptr + row * D + offs, mask=mask, other=0.0)
    tl.store(unified_ptr + loc * D + offs, vals, mask=mask)
```

评论区精华

代码审查中，gemini-code-assist 提出了多个高优先级问题：

- 在 paged_decode 和 paged_prefill 内核中，当 slot 为 -1（哨兵）时，指针算术会产生负偏移，可能触发 AMD GPU 页面错误。
- 在 runtime.py 中，当压缩窗口大小 Wc 不是 2 的幂时，BLOCK 可能大于实际长度，导致越界读取。
- 在 deepseek_v4.py 中，使用均匀的 repeat_interleave 来映射 token 到请求可能在 prefill 时产生错误。
- 在 split-K 路径上，每次 decode 都分配 m_partial / l_partial 等张量，在 CUDA Graph 外可能造成开销。 作者 1am9trash 回应：已在 paged_decode 和 paged_prefill 中为 slot 添加了 maximum(slot, 0) 保护，并在 runtime.py 中为 j 索引添加了 minimum(j, Wc - 1) 保护，且经过端到端测试未观察到 fault。关于 repeat_interleave 和分配 overhead 的问题未进一步回复。此外，amd-bot多次指出新路径在CI中未被执行（无测试设置该环境变量），建议不要仅凭绿色 CI 合并。但作者确认 NVIDIA V4 测试通过后合并。
- 负指针安全：slot=-1 时的 GPU page fault 风险 (correctness): 作者 1am9trash 确认测试中未触发错误，但为了防御性编程，在所有相关位置添加了 tl.maximum(slot, 0) 或 tl.minimum(j, Wc-1) 保护。已在最终代码中体现。
- Wc 非 2 的幂时的越界读取 (correctness): 作者添加了 tl.minimum(j, Wc-1) 保护，已修复。
- repeat_interleave 在 prefill 时可能错误映射 (correctness): 作者未直接回复，但 PR 已合并，可能认为当前 decode 下 uniform 假设成立，或已通过其他方式保证。未看到进一步修改。
- CUDA Graph 外 split-K 分配的 overhead (performance): 作者未回复，该问题未解决。

风险与影响

- 风险：

- GPU page fault 风险：尽管添加了 clamp，若其他边界情况未覆盖仍可能触发（如零长度的 token 导致 indptr 相同但索引可能无效）。
- 测试覆盖缺失：新路径约 1800 行内核代码在 CI 中从未执行，仅通过了 NVIDIA 的默认路径测试。AMD 硬件上的正确性和稳定性依赖 nightly 验证。
- 性能退化可能：split-K 路径每次 forward 分配中间张量，在非 CUDA Graph 场景下可能产生 overhead；同时统一池的每层全量零初始化可能增加显存占用。
- 兼容性：新路径完全门控，默认路径不变，不影响现有用户。但若用户启用混合环境变量，需确保 kernel 与 aiter 库版本兼容。

- 影响：
  - 用户影响：AMD 用户可通过设置环境变量选择启用新后端，期望在 DeepSeek-V4 推理中获得更高吞吐，但需自行验证稳定性。NVIDIA 用户不受影响。
  - 系统影响：代码体积增加约 2400 行，主要分布在 unified_kv_kernels 目录；新增 DeepSeekV4UnifiedKVPool 类作为可选组件。
  - 团队影响：AMD 团队需要负责该路径的后续维护和 bug 修复；社区贡献者也可参与。
  - 风险标记：核心路径变更但缺少测试覆盖，GPU page fault 风险（已部分修复），split-K 路径分配 overhead，默认禁用降低风险

关联脉络

- 暂无明显关联 PR