PR #23038 完整报告

sgl-project/sglang

[KDA] Fuse gate+cumsum and reuse chunk index for KDA

合并时间：2026-04-21 17:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23038

执行摘要

- 一句话：融合 KDA 的 gate+cumsum 操作并重用 chunk index，提升内核性能 2.2-2.65 倍和端到端吞吐量 6-11%。
- 推荐动作：该 PR 值得精读，特别是 kda_gate_chunk_cumsum 内核的实现展示了如何通过融合计算减少内存往返，以及 chunk index 重用优化避免了重复预处理。关注设计权衡（如简化路径、测试覆盖）和性能提升技巧。

功能与动机

优化 KDA 内核性能，减少内存流量和重复计算。PR body 中说明：'Optimize KDA kernel with fusing gate+cumsum and reusing chunk index.'，旨在通过融合操作消除一个内核启动和一次全局内存往返（HBM 流量从 136MB 降至 68MB），并通过重用 chunk index 避免在 varlen 模式下 5-6 次冗余的 prepare_chunk_indices 调用，从而提升整体效率。

实现拆解

1. 新增融合内核kda_gate_chunk_cumsum：在python/sglang/srt/layers/attention/fla/kda.py 中新增 Triton 内核函数 kda_gate_chunk_cumsum 和 kda_gate_chunk_cumsum_vector_kernel，整合 gate 激活（公式 -exp(A_log) * softplus(raw_g + dt_bias)）和 chunk-local cumulative sum，减少内存往返。内核使用 autotune 配置（BS_LIST 基于共享内存检测优化）。
2. 修改 KDA 主函数以使用融合内核：在 kda.py 的 chunk_kda_fwd 函数中，添加 A_log、dt_bias 等参数，根据是否提供 A_log 选择调用 kda_gate_chunk_cumsum（融合路径）或 chunk_local_cumsum（传统路径），并预计算 chunk_indices 传递给下游函数以避免重复计算。
3. 更新模型层适配：修改 python/sglang/srt/models/kimi_linear.py，移除对 fused_kda_gate 的导入和调用，在 prefill 模式中传递 raw gate（未激活）给 chunk_kda_fwd，让融合内核处理 gate 激活。
4. 添加基准测试和单元测试：新增 benchmark/bench_linear_attention/bench_fused_gate_cumsum.py，包含 make_inputs、run_ref、run_fused 等函数，用于性能对比和验证；扩展 test/registered/attention/test_kda_kernels.py，新增 TestKDAGateChunkCumsum 测试类，覆盖 varlen、bias 等多种场景，确保正确性（max_diff<2e-4）。

5. 其他相关文件调整：更新 cumsum.py、chunk_delta_h.py、chunk_intra.py 等文件，添加 chunk_indices 参数并优化逻辑，确保整个调用链支持 index 重用。

关键文件：

- benchmark/bench_linear_attention/bench_fused_gate_cumsum.py（模块 基准测试；类别 source；类型 benchmark；符号 make_inputs, run_ref, run_fused, verify_correctness）：新增的基准测试文件，用于对比融合与分离路径的性能，验证优化效果，包含关键函数如 run_ref 和 run_fused。
- python/sglang/srt/layers/attention/fla/kda.py（模块 注意力内核；类别 source；类型 core-logic；符号 softplus_fwd, kda_gate_chunk_cumsum_vector_kernel, kda_gate_chunk_cumsum, chunk_kda_fwd）：核心实现文件，新增融合内核 kda_gate_chunk_cumsum 并修改 chunk_kda_fwd 以支持融合和 chunk index 重用，影响 KDA 主路径。
- test/registered/attention/test_kda_kernels.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestKDAGateChunkCumsum, _ref_gate_cumsum, _run_case, test_varlen_with_bias）：扩展的单元测试文件，新增 TestKDAGateChunkCumsum 类验证融合内核的正确性，覆盖多种场景（varlen、bias 等）。

关键符号：kda_gate_chunk_cumsum, chunk_kda_fwd, softplus_fwd, chunk_local_cumsum, prepare_chunk_indices

关键源码片段

benchmark/bench_linear_attention/bench_fused_gate_cumsum.py

新增的基准测试文件，用于对比融合与分离路径的性能，验证优化效果，包含关键函数如 run_ref 和 run_fused。

```
def run_ref(inp):
    """Separate path: torch gate activation -> chunk_local_cumsum."""
    raw_g = inp["raw_g"] # [1, T, H, K]
    A_log = inp["A_log"] # [H]
    dt_bias = inp["dt_bias"] # [H*K]
    cu_seqlens = inp["cu_seqlens"]
    H, K = inp["H"], inp["K"]

    # Step 1: gate activation using torch ops
    g_float = raw_g.float()
    if dt_bias is not None:
        g_float = g_float + dt_bias.float().view(1, 1, H, K) # 添加 bias
    g_activated = -torch.exp(
        A_log.float().view(1, 1, H, 1)
    ) * torch.nn.functional.softplus(g_float) # 计算激活后的 gate

    # Step 2: chunk-local cumsum
    chunk_indices = prepare_chunk_indices(cu_seqlens, CHUNK_SIZE)
    g_cumsum = chunk_local_cumsum(
        g_activated,
```

```
        chunk_size=CHUNK_SIZE,
        cu_seqlens=cu_seqlens,
        chunk_indices=chunk_indices,
    ) # 执行 cumsum
    return g_cumsum


def run_fused(inp):
    """Fused path: kda_gate_chunk_cumsum (single kernel)."""
    raw_g = inp["raw_g"]
    A_log = inp["A_log"]
    dt_bias = inp["dt_bias"]
    cu_seqlens = inp["cu_seqlens"]

    chunk_indices = prepare_chunk_indices(cu_seqlens, CHUNK_SIZE)
    g_cumsum = kda_gate_chunk_cumsum(
        raw_g,
        A_log=A_log,
        chunk_size=CHUNK_SIZE,
        dt_bias=dt_bias,
        cu_seqlens=cu_seqlens,
        chunk_indices=chunk_indices,
    ) # 调用融合内核，一次性完成 gate 激活和 cumsum
    return g_cumsum
```

python/sglang/srt/layers/attention/fla/kda.py

核心实现文件，新增融合内核 kda_gate_chunk_cumsum 并修改 chunk_kda_fwd 以支持融合和 chunk index 重用，影响 KDA 主路径。

```
def chunk_kda_fwd(
    q: torch.Tensor,
    k: torch.Tensor,
    v: torch.Tensor,
    g: torch.Tensor,
    beta: torch.Tensor,
    scale: float,
    initial_state: torch.Tensor,
    initial_state_indices: torch.Tensor,
    cu_seqlens: Optional[torch.LongTensor] = None,
    A_log: Optional[torch.Tensor] = None, # 新增参数：gate 激活的 log-scale
    dt_bias: Optional[torch.Tensor] = None, # 新增参数：gate 的 bias
    lower_bound: Optional[float] = None,
):
    chunk_size = 64
    # 预计算 chunk_indices，避免下游函数重复计算
    chunk_indices = (
        prepare_chunk_indices(cu_seqlens, chunk_size) if cu_seqlens is not None else None
    )
```

```
if A_log is not None:
    # 融合路径：gate 激活 + cumsum
    g = kda_gate_chunk_cumsum(
        g,
        A_log=A_log,
        dt_bias=dt_bias,
        lower_bound=lower_bound,
        chunk_size=chunk_size,
        cu_seqlens=cu_seqlens,
        chunk_indices=chunk_indices, # 传递预计算的 index
    )
else:
    # 传统路径：g 已由调用者激活，仅执行 cumsum
    g = chunk_local_cumsum(
        g,
        chunk_size=chunk_size,
        cu_seqlens=cu_seqlens,
        chunk_indices=chunk_indices, # 传递预计算的 index
    )
# 后续 KDA 计算逻辑保持不变
...
```

评论区精华

1. 设计简化：reviewer kaixih 指出 chunk_kda_fwd 中三条路径过于复杂，建议简化为基于 A_log 的条件分支。作者采纳，修改后逻辑更清晰。
2. 测试覆盖：kaixih 询问新内核的单元测试，作者回应 'Added unit test.'，在测试文件中新增了 TestKDAGateChunkCumsum 类进行验证。
3. 优化解释：reviewer rainj-me 质疑预计算 chunk_indices 是否冗余，作者解释这是有意优化，避免下游函数（如 kda_gate_chunk_cumsum、chunk_local_cumsum）重复调用 prepare_chunk_indices，从而减少开销。
4. 代码清理：kaixih 建议移除不再使用的 fused_kda_gate 函数，作者回应 'Removed.'，并从 kimi_linear.py 中删除相关导入和调用。

- 简化 chunk_kda_fwd 中的逻辑路径 (design): 作者采纳建议，修改为 if-else 结构，使代码更清晰。
- 为新融合内核添加单元测试 (testing): 作者回应 'Added unit test.'，在测试文件中新增了 TestKDAGateChunkCumsum 类进行验证。
- chunk index 重用优化的必要性 (performance): 作者解释这是有意优化，避免下游函数（如 kda_gate_chunk_cumsum、chunk_local_cumsum）重复调用 prepare_chunk_indices，减少开销。

风险与影响

- 风险：

1. 正确性风险：融合内核可能引入数值误差，但测试显示 max_diff=1.53e-04、rel_diff=4.13e-06，在可接受范围内；需确保所有边界情况（如 varlen、无 bias）被测试覆盖。
2. 性能回归风险：新内核的 autotune 配置（BS_LIST）依赖共享内存检测，在 AMD 等平台需验证适应性；从提交看已调整以支持多平台。
3. 兼容性风险：多个函数（如 recompute_w_u_fwd、chunk_gla_fwd_o_gk）添加了 chunk_indices 参数，需确保所有调用点更新，相关文件（如 chunk_delta_h.py）已适配。
4. 维护风险：新增内核和逻辑可能增加代码复杂度，但 review 中已简化设计并移除旧函数。

- 影响：
  1. 用户影响：端到端 KDA 预填充吞吐量提升 6-11%，尤其在长序列（T_total≥4096）和大批次下收益显著，峰值吞吐量从 8637 tok/ms 增至 9520 tok/ms，改善推理延迟。
  2. 系统影响：内核级内存流量减半，减少 GPU 内存带宽压力；chunk index 重用降低 CPU-GPU 传输和计算开销，提升整体系统效率。
  3. 团队影响：代码更简洁（移除 fused_kda_gate），但需维护新内核；设计模式（融合操作、index 重用）可为其他优化提供参考。 - 风险标记：核心路径变更，数值精度风险，接口变更

关联脉络

- PR #22544 [Score API] Add Multi-Item Scoring with pre-computed delimiter indices: 类似地采用了预计算索引优化来消除 GPU 扫描，提升性能，与本 PR 的 chunk index 重用策略有共通之处。
- PR #23315 Opt-in strip of thinking tokens from radix cache: 同样涉及缓存和性能优化，展示了仓库中对内存和计算效率的持续改进趋势。