PR #27488 完整报告

sgl-project/sglang

[KDA] Add CuteDSL Prefill Kernel on SM100

合并时间：2026-06-10 21:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27488

执行摘要

- 一句话：新增 KDA SM100 CuteDSL 预填充内核
- 推荐动作：建议深入阅读 __init__.py 中的数值修复方案（sub-chunk 归一化处理 per-channel gate）和 _kda_workspace 的缓存设计，是高性能算子实现的优秀示例。同时了解如何通过 TMA 和 MMA 编程在 Blackwell 上实现高效 sequence-parallel 内核。

功能与动机

KDA 的 per-channel decay gate 导致 fp32 exp 在真实模型参数下溢出（exp 参数最大可达 6e4），现有 Triton 实现输出 NaN。同时 CuteDSL 内核本身快速但受 host overhead（每次调用重新分配约 200MB 张量）瓶颈，导致整体比 Triton 慢。需要解决数值稳定性并消除 host 开销。

实现拆解

1. 数值修复：在 kda_blackwell/__init__.py 的 chunk_kda_cutedsl 中，将两个会溢出的 operand（kL, qg2）替换为通过 sub-chunk 归一化 FLA 内核 chunk_kda_scaled_dot_kkt_fwd 计算的 gated 矩阵，再通过恒等右 operand 注入不变 CuteDSL MMA，确保所有 exp 指数≤0。
2. 性能优化：引入模块级 _KDA_WS 可重用工作空间，按 (Hv, K, V, device, dtype, stream) 键控，避免每次调用重新分配清零；eye 和元数据仅在 cu_seqlens 对象变化时重算，无同步。
3. Extend 修复：在 kda_cutedsl.py 的 CuteDSLKDAKernel.extend 中，将 g 和 beta 按 q 的实际 token 数裁剪，修复 cuda-graph padding 下形状不匹配导致的崩溃。
4. 后端集成：在 kda_backend.py 中移除临时的 numerically unstable 防护，使 CuteDSL 预填充默认启用。
5. 测试与基准：新增 test_kda_prefill_cutedsl.py（含真实门控测试）和 bench_kda_prefill_cutedsl.py，验证正确性和性能。

关键文件：

- python/sglang/srt/layers/attention/linear/kernels/kda_blackwell/kernel_kkt_inv_uw.py（模块 KKT 逆内核；类别 source；类型 core-logic；符号 Sm100KdaChunkUWKernel, init, _make_tma_args, call）：新增 SM100 KKT 逆 +U/W 专有内核，实现 per-channel 门控下的矩阵求逆和 U/W 计算，是预填充管线的核心计算步。

- python/sglang/srt/layers/attention/linear/kernels/kda_blackwell/kernel_h.py（模块 状态更新核；类别 source；类型 core-logic；符号 Sm100KdaChunkHKernel, init, _make_bf16_tma_args, _make_h_tma_args）：新增 SM100 KDA chunk recurrent-state 更新内核，处理 per-column 状态衰减，使用预缩放的 kg 张量。
- python/sglang/srt/layers/attention/linear/kernels/kda_blackwell/kernel_o.py（模块 输出内核；类别 source；类型 core-logic；符号 Sm100KdaChunkOKernel, init, _make_bf16_tma_args, _make_h_tma_args）：新增 SM100 KDA 输出内核，使用预缩放 qg/qg2/kg 张量，无需 g_cu 输入。
- python/sglang/srt/layers/attention/linear/kernels/kda_blackwell/__init__.py（模块 预填充管道；类别 source；类型 dependency-wiring；符号 prepare_metadata, _kda_workspace, chunk_kda_cutedsl）：编排整个 CuteDSL prefill 管道：prepare_metadata 生成 chunk 索引，_kda_workspace 提供可重用工作空间，chunk_kda_cutedsl 实现数值修复后的 prefill 入口。
- benchmark/bench_linear_attention/bench_kda_prefill_cutedsl.py（模块 基准测试；类别 source；类型 dependency-wiring；符号 _l2norm, kda_flops, kda_bytes, make_inputs）：提供 CuteDSL vs Triton 的性能和正确性基准测试，验证加速比和数值误差。
- python/sglang/srt/layers/attention/linear/kernels/kda_blackwell/prologue.py（模块 融合序言；类别 source；类型 core-logic；符号 _kda_prologue_kernel, kda_prologue）：Triton 融合序言，计算 per-chunk cumsum 和五组预缩放张量，弥合 per-channel 门控与 CuteDSL MMA 的接口。
- python/sglang/srt/layers/attention/linear/kernels/kda_cutedsl.py（模块 后端集成；类别 source；类型 dependency-wiring；符号 _is_blackwell, init, _ensure_extend_loaded, extend）：后端集成：CuteDSLKDAKernel 现在支持 extend（prefill），并修复 cuda-graph padding 导致的形状不匹配。
- python/sglang/srt/layers/attention/linear/kda_backend.py（模块 后端适配；类别 source；类型 dependency-wiring）：移除临时数值不稳定防护，使 CuteDSL prefill 默认启用。
- test/registered/attention/test_kda_prefill_cutedsl.py（模块 回归测试；类别 test；类型 test-coverage；符号 _l2norm, test_kda_chunk_cutedsl_correctness, test_kda_chunk_cutedsl_internal_gate_activation, test_kda_chunk_cutedsl_realistic_gate）：新增回归测试，包含真实门控测试（可检测之前未发现的数值溢出），以及 varlen 正确性和内部门控激活测试。

关键符号：chunk_kda_cutedsl, prepare_metadata, _kda_workspace, kda_prologue, Sm100KdaChunkUWKernel.call, Sm100KdaChunkHKernel.call, Sm100KdaChunkOKernel.call, CuteDSLKDAKernel.extend, CuteDSLKDAKernel._ensure_extend_loaded

关键源码片段

python/sglang/srt/layers/attention/linear/kernels/kda_blackwell/kernel_kkt_inv_uw.py

新增 SM100 KKT 逆 +U/W 专有内核，实现 per-channel 门控下的矩阵求逆和 U/W 计算，是预填充管线的核心计算步。

```
# File: kernel_kkt_inv_uw.py
# SPDX-License-Identifier: Apache-2.0
# KDA SM100 KKT-inverse + U/W kernel (gate folded outside into pre-scaled keys)

class Sm100KdaChunkUWKernel:
    def __call__(
        self,
        KL, # k * exp(g_cu - g_cu_last) [T, Hv, K]
        KR, # k * exp(g_cu_last - g_cu) [T, Hv, K]
        KG, # k * exp(g_cu) [T, Hv, K]
        V, U, W, beta, cu_seqlens, chunk_indices, total_chunks, num_sms, stream
    ):
        tma_g2s = cpasync.CopyBulkTensorTileG2SOp()
        tma_s2g = cpasync.CopyBulkTensorTileS2GOp()
        KL_args = self._make_tma_args(KL, self.K_dim, self.num_stages, tma_g2s)
        KR_args = self._make_tma_args(KR, self.K_dim, self.num_stages, tma_g2s)
        KG_args = self._make_tma_args(KG, self.K_dim, self.num_stages, tma_g2s)
        V_args = self._make_tma_args(V, self.V_dim, self.num_stages, tma_g2s)
        U_args = self._make_tma_args(U, self.V_dim, 1, tma_s2g)
        W_args = self._make_tma_args(W, self.K_dim, 1, tma_s2g)
        grid = (num_sms // self.Hv, self.Hv, 1)
        block = (self.num_warps * 32, 1, 1)
        self.kernel(
            KL_args, KR_args, KG_args, V_args, U_args, W_args,
            beta, cu_seqlens, chunk_indices, total_chunks,
        ).launch(grid=grid, block=block, stream=stream)
```

python/sglang/srt/layers/attention/linear/kernels/kda_blackwell/__init__.py

编排整个 CuteDSL prefill 管道：prepare_metadata 生成 chunk 索引，_kda_workspace 提供可重用工作空间，chunk_kda_cutedsl 实现数值修复后的 prefill 入口。

```
# File: __init__.py
# KDA SM100/Blackwell CuteDSL prefill pipeline

def prepare_metadata(cu_seqlens, chunk_size=64):
    cs = cu_seqlens.to(torch.int64)
    seqlens = cs[1:] - cs[:-1]
    nchunks = (seqlens + chunk_size - 1) // chunk_size
    chunk_offsets = torch.zeros(nchunks.numel()+1, dtype=torch.int32, device=dev)
    chunk_offsets[1:] = nchunks.cumsum(0).to(torch.int32)
    total = int(chunk_offsets[-1].item())
    seq_id = torch.repeat_interleave(torch.arange(n, device=dev), nchunks)
    local = torch.arange(total, device=dev) - chunk_offsets[seq_id].to(torch.int64)
    chunk_indices = torch.stack([seq_id.to(torch.int32), local.to(torch.int32)], dim=1)
    return chunk_indices, chunk_offsets, torch.tensor([total], device=dev), total

_KDA_WS = {}
def _kda_workspace(q, T, Hv, K, V, cu_seqlens):
    stream = torch.cuda.current_stream(device=q.device).cuda_stream
```

```
    key = (Hv, K, V, q.device, q.dtype, stream)
    ws = _KDA_WS.get(key)
    if ws is None or ws['cu'] is not cu_seqlens:
        ci, co, tcs, total = prepare_metadata(cu_seqlens)
        # ... 分配或重用 scratch
    return ws
```

python/sglang/srt/layers/attention/linear/kernels/kda_cutedsl.py

后端集成：CuteDSLKDAKernel 现在支持 extend（prefill），并修复 cuda-graph padding 导致的形状不匹配。

```
# File: kda_cutedsl.py (CuteDSLKDAKernel class)
def extend(self, q, k, v, g, beta, *, ssm_states, cache_indices, query_start_loc, A_log=None, dt_
bias=None, lower_bound=None, **kwargs):
    head_k_dim = k.shape[-1]
    self._ensure_extend_loaded(head_k_dim)
    # L2 normalize and convert to bf16
    q_n = self._l2norm_fn(q[0].contiguous()).to(torch.bfloat16)
    k_n = self._l2norm_fn(k[0].contiguous()).to(torch.bfloat16)
    v_in = v[0].contiguous().to(torch.bfloat16)
    # Trim g and beta to real token count (fix for cuda-graph padding)
    num_tokens = q_n.shape[0]
    g_in = g[0][:num_tokens]
    beta_in = beta[0][:num_tokens]
    # Call the cutedsl prefill pipeline
    o, ht = self._extend_fn(q_n, k_n, v_in, g_in, beta_in, ...)
    return o[None] # restore batch dim
```

评论区精华

- gemini-code-assist[bot]（高优先级）：指出全局工作空间 _KDA_WS 在多个 CUDA 流并发时存在数据竞争，建议将流 ID 加入键。 → 提交者已采纳。
- gemini-code-assist[bot]（中优先级）：建议使用 int32 创建 tok 避免 int64 转换，消除不必要的 .long() 开销。 → 提交者已采纳并微小调整。
- BBuf：要求在 kernel_h.py 头注释中添加仓库链接。 → 提交者已修正注释。
- 全局工作空间多流并发 (correctness): yuan-luo 采纳建议，在 commit 中将当前 CUDA 流 ID 加入键。
- int32 tok 避免 int64 转换 (performance): yuan-luo 应用了建议并稍作调整。
- kernel_h.py 缺少仓库链接 (documentation): yuan-luo 修订了注释。

风险与影响

- 风险：
  - 硬件限制：CuteDSL 内核仅 SM100+(Blackwell) 支持，其他 GPU 自动回退到 Triton，可能因未启用新路径而无影响。

    - 工作空间内存：_KDA_WS 按配置缓存各种张量，可能在动态 shape 场景下保留大量内存，但通过键匹配可及时重用，未释放时不增常驻内存。
    - 多流并发：初始实现键未包含流，已被修复，经查无竞争。
    - 数值精度：与 token-by-token 基准相比，o_err 为 4.88e-4，全有限，与无修复时的 NaN 相比已正确。
- 影响：
    - 用户影响：使用 Kimi-Linear 模型在 Blackwell GPU 上的推理用户将获得正确结果和显著加速 (1.08x-1.52x)。其他模型或 GPU 无影响（回退）。
    - 系统影响：增加约 4.5k 行代码，主要在内核和测试目录。构建需要 CuteDSL 和 cuTENSOR 支持，已在 ci 覆盖。
    - 团队影响：为后续 KDA decode 内核或更多 linear attention 的 Blackwell 优化提供了参考模式。
    - 风险标记：Blackwell 限定，工作空间内存，多流并发

关联脉络

- 暂无明显关联 PR