PR #31904 完整报告

sgl-project/sglang

[KDA] Fix mixed exponent bases in Triton chunk prefill

合并时间：2026-07-22 16:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31904

执行摘要

- 一句话：修复 KDA chunk prefill 中 exp2/exp 混合导致的衰减错误
- 推荐动作：该 PR 值得精读，因为它清晰地展示了如何通过规范指数域来修复不匹配 bug，同时采用编译选项保证兼容性。设计决策（如 gk_scale 避免物化、编译时 use_exp2）可作为类似问题的参考。

功能与动机

KDA 定义每个通道的衰减门在自然对数空间，累计门 $G_t = \sum g_s$，从位置 j 到 i 的衰减应为 $\exp(G_i-G_j)$。但部分 intra-chunk kernel 使用 exp2 直接计算，而其他路径使用自然 exp，导致混合指数基底，实际计算的衰减弱于理论值。

实现拆解

1. 在 KDA 的累计门生产处（kda_gate_chunk_cumsum 和 chunk_local_cumsum）加入 RCP_LN2 = 1.4426950216293335（log2(e)），将自然对数门转换为 log2 空间。
2. 所有 KDA chunk kernel（scaled KKT/QK 构建、独立和融合的 W/U/KG 重计算、chunk 内融合求解 / 重计算、状态更新、输出构建）统一改用 exp2。
3. 在共享状态 kernel chunk_gated_delta_rule_fwd_h 中添加编译选项 use_exp2，默认 False 保持 GDN 原行为，KDA 传入 True。
4. 在 chunk_kda_scaled_dot_kkt_fwd 中添加 gk_scale 参数，避免物化转换后的门张量，直接在 kernel 加载时乘以 RCP_LN2。
5. 新增独立 PyTorch 递归参考实现，验证固定长度和变长场景下，chunk prefill 的输出和最终状态与理论值误差小于 1%。

关键文件：

- test/registered/attention/test_kda_kernels.py（模块 回归测试；类别 test；类型 test-coverage；符号 TestKDAChunkExponentDomain, _naive_recurrent, _relative_rmse, test_chunk_prefill_matches_natural_exp_recurrence）：新增 TestKDAChunkExponentDomain 测试类，覆盖固定长度、变长、融合 / 非融合多条代码路径，回归验证修复的正确性。
- python/sglang/kernels/ops/attention/fla/kda.py（模块 KDA 核心；类别 infra；类型 core-logic；符号 chunk_kda_scaled_dot_kkt_fwd, chunk_kda_scaled_dot_kkt_fwd_kernel_intra_sub_inter,

chunk_kda_scaled_dot_kkt_fwd_kernel_intra_sub_intra）：核心修改文件：定义 RCP_LN2，修改 scaled_dot_kkt kernel 和入口函数，统一使用 exp2，添加 gk_scale 参数。

- python/sglang/kernels/ops/attention/fla/chunk_delta_h.py（模块 状态 kernel；类别 infra；类型 core-logic；符号 chunk_gated_delta_rule_fwd_kernel_h_blockdim64, chunk_gated_delta_rule_fwd_h）：共享状态 kernel 添加 use_exp2 编译选项，使 KDA 使用 exp2 而 GDN 保持不变。
- python/sglang/kernels/ops/attention/fla/chunk_intra.py（模块 Intra kernel；类别 infra；类型 core-logic；符号 chunk_kda_fwd_kernel_inter_solve_fused, chunk_kda_fwd_intra）：KDA intra kernel（对角化、重计算、融合求解）全部将 exp 替换为 exp2，保持一致性。
- python/sglang/srt/hardware_backend/xpu/kernels/fla/chunk_delta_h.py（模块 XPU 适配；类别 source；类型 dependency-wiring；符号 chunk_gated_delta_rule_fwd_kernel_h_blockdim64_k_loop, chunk_gated_delta_rule_fwd_h）：XPU 适配：同步 use_exp2 编译选项，与 CUDA 版本对齐。

关键符号：kda_gate_chunk_cumsum, chunk_local_cumsum, chunk_kda_scaled_dot_kkt_fwd, chunk_kda_fwd_intra, chunk_gated_delta_rule_fwd_h

关键源码片段

test/registered/attention/test_kda_kernels.py

新增 TestKDAChunkExponentDomain 测试类，覆盖固定长度、变长、融合 / 非融合多条代码路径，回归验证修复的正确性。

```
@unittest.skipIf(not torch.cuda.is_available(), "Test requires CUDA")
class TestKDAChunkExponentDomain(CustomTestCase):
    """Guard KDA prefill against mixing natural-log gates with exp2 kernels."""

    @staticmethod
    def _naive_recurrent(q, k, v, g, beta, initial_state, lengths):
        # Pure-PyTorch 逐 token 递归，作为无 bug 参考
        q, k, v, g, beta = (tensor.float() for tensor in (q, k, v, g, beta))
        scale = q.shape[-1] ** -0.5
        output = torch.empty_like(v)
        final_state = initial_state.float().clone()

        offset = 0
        for seq_idx, length in enumerate(lengths):
            state = final_state[seq_idx]
            for i in range(offset, offset + length):
                state = state * g[0, i].exp().unsqueeze(-2) # 自然 exp
                residual = v[0, i] - torch.einsum("hvk,hk->hv", state, k[0, i])
                state = state + torch.einsum(
                    "hv,hk->hvk",
                    residual * beta[0, i, :, None],
                    k[0, i],
                )
```

```
                output[0, i] = (
                    torch.einsum("hvk,hk->hv", state, q[0, i]) * scale
                )
            final_state[seq_idx] = state
            offset += length
        return output, final_state

    @torch.inference_mode()
    def test_chunk_prefill_matches_natural_exp_recurrence(self):
        device = get_device()
        dtype = torch.bfloat16
        num_heads, head_dim = 2, 64

        # 测试组合：(lengths, use_varlen, fuse_gate)
        cases = (
            ([129], False, False), # 固定长度，pre-activated gate
            ([15, 16, 17, 63, 65], True, True), # varlen，fused raw gate
            ([2] * 129, True, False), # 强制非融合路径（_small_grid=False）
        )
        for lengths, use_varlen, fuse_gate in cases:
            with self.subTest(lengths=lengths, use_varlen=use_varlen, fuse_gate=fuse_gate):
                # 构造输入 ...（省略具体构造）
                # 运行 chunk_kda 与 naive_recurrent，断言 relative_rmse < 0.01
```

python/sglang/kernels/ops/attention/fla/kda.py

核心修改文件：定义 RCP_LN2，修改 scaled_dot_kkt kernel 和入口函数，统一使用 exp2，添加 gk_scale 参数。

```
# 位于 python/sglang/kernels/ops/attention/fla/kda.py

# log2(e) 的 FP32 近似值，用于将自然对数门转换到 log2 空间
RCP_LN2 = 1.4426950216293335

def chunk_kda_scaled_dot_kkt_fwd(
    q, k, gk=None, beta=None, scale=None,
    gk_scale=1.0, # <-- 新增参数，避免物化转换张量
    cu_seqlens=None, chunk_indices=None, output_dtype=None,
):
    # ... 省略形状推导
    # 将 gk_scale 传递给 Triton kernel
    grid = ...
    chunk_kda_scaled_dot_kkt_fwd_kernel_intra_sub_inter[grid](
        ..., gk_scale=gk_scale, ...
    )

# 在 kernel 内部（以 _intra_sub_inter 为例）:
# ...
b_gn = (
    tl.load(g + ...) * gk_scale # 加载时转换
```

```
)
# 所有 gate 相关的 exp 替换为 exp2
b_k = tl.load(...) * exp2(b_g - b_gn[None, :])
```

评论区精华

Reviewer kaixih 建议在 chunk_kda_scaled_dot_kkt_fwd 中添加 gk_scale 参数，直接加载时乘以 RCP_LN2，避免物化完整的转换张量。同时建议在测试中添加非融合路径的覆盖案例（force non-fused path），通过大量 chunk 使 _small_grid=False 来触发独立对角化和重计算 kernel。Author yuan-luo 采纳了两项建议，体现在第二个和第三个 commit 中。

- 使用 gk_scale 避免物化转换张量 (design): Author 采纳，添加 gk_scale 参数并传递至 Triton kernel，Blackwell 后端同样使用。
- 增加非融合路径的测试覆盖 (testing): Author 添加该案例，并在测试中说明其目的。

风险与影响

- 风险：修改集中在 KDA chunk prefill 路径，解码路径不受影响；GDN 通过 use_exp2 编译选项隔离，默认行为不变；Blackwell 后端通过 gk_scale 同步适配。但未进行端到端模型精度基准测试，kernel 级回归测试覆盖了核心路径，可能遗漏边界条件。XPU 适配代码已同步调整但无法在 CI 中自动验证。
- 影响：影响使用 KDA chunk prefill 的模型（如 DeepSeek V2/V3 等），能纠正精度错误，提升生成质量。用户无需任何配置变更。不影响显式单步解码或 GDN。预期性能中性或略有提升。
- 风险标记：KDA 核心路径变更，缺少端到端模型精度基准，XPU 路径需人工验证

关联脉络

- 暂无明显关联 PR