PR #31986 完整报告

sgl-project/sglang

[Perf] Stack dspark dense draft per-layer ctx KV projection into one GEMM

合并时间：2026-07-22 08:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31986

执行摘要

- 一句话：堆叠 DSpark 各层 KV 投影为单次 GEMM，减少推测解码开销
- 推荐动作：建议精读该 PR，特别是 _stacked_ctx_kv_params 的兼容性检查和缓存策略，以及 write_target_hidden_kv 中的安全回退机制。该模式可推广到其他需要逐层融合的场景。测试设计值得学习——直接对比数值而非依赖端到端精度，确保优化不破坏代数等价性。

功能与动机

在推测解码热路径的 write_target_hidden_kv 中，每层独立执行 KV 投影 GEMM、k-norm 和 RoPE，存在大量冗余。由于各层输入 ctx_hidden 相同，将 N 个并行计算合并为一个在数学上完全等价。PR 旨在通过融合减少 GPU 调用次数，提升推理吞吐。

实现拆解

1. 新增兼容性检测与参数堆叠方法(_stacked_ctx_kv_params)：遍历 self.layers 中各层 self_attn，调用 can_dflash_slice_qkv_weight 确认 qkv_proj 未量化（可切片），并检查各层 k_norm.variance_epsilon 是否一致、偏置存在性是否统一。满足条件时，从每层 qkv_proj.weight 中切片出 KV 部分（kv_slice），连同偏置和 k_norm.weight 分别收集，最终拼接为 weight、bias（可选）和 k_norm_weight（float32 堆叠），存入缓存字典。否则返回 None 触发回退。
2. 新增融合投影方法(_project_ctx_kv_stacked)：利用拼接后的权重对 ctx_hidden 执行一次 GEMM 得到所有层的 K/V 输出，然后按层拆分，对每个头执行批量化的 RMSNorm（使用缓存的 k_norm_weight 和 eps），再通过共享的 RoPE 计算旋转位置编码。最终返回 k_all（列表，每层 [batch*seq, num_kv_heads, head_dim]）和 v_all。
3. 修改主入口(write_target_hidden_kv)：先调用 _stacked_ctx_kv_params；若返回非 None，则调用 _project_ctx_kv_stacked 得到合并的 K/V，然后在循环中直接按层索引分配到 KV缓存；否则保留原有逐层计算逻辑（kv_proj_only→apply_k_norm→apply_k_rope）。
4. 添加数值一致性测试(test/registered/spec/dspark/test_dspark_stacked_ctx_kv_parity.py)：使用随机模拟的 DFlashAttention（每层权重不同，层序可辨）分别计算逐层参考输出和堆叠输出，在 fp16/bf16 精度下逐层比较 K/V 的张量接近性（rtol=5e-3/2e-2）。额外验证量化、epsilon 不一致、偏置不一致三种场景下 _stacked_ctx_kv_params 返回 None。

关键文件：

- python/sglang/srt/models/dspark.py（模块 草稿模型；类别 source；类型 core-logic；符号 _stacked_ctx_kv_params, _project_ctx_kv_stacked, write_target_hidden_kv）：包含核心逻辑：新增堆叠参数检测（_stacked_ctx_kv_params）、融合投影方法（_project_ctx_kv_stacked）以及 write_target_hidden_kv 的改造。是性能提升的关键文件。
- test/registered/spec/dspark/test_dspark_stacked_ctx_kv_parity.py（模块 测试；类别 test；类型 test-coverage；符号 TestDSparkStackedCtxKvParity, _check_parity, _per_layer_reference, _make_attn）：新增的数值一致性测试文件，是 PR 正确的关键保障，覆盖正常堆叠路径和三种回退条件。

关键符号：_stacked_ctx_kv_params, _project_ctx_kv_stacked, write_target_hidden_kv, _check_parity

关键源码片段

python/sglang/srt/models/dspark.py

包含核心逻辑：新增堆叠参数检测（_stacked_ctx_kv_params）、融合投影方法（_project_ctx_kv_stacked）以及 write_target_hidden_kv 的改造。是性能提升的关键文件。

```
import torch
import torch.nn.functional as F
from sglang.srt.speculative.dflash_utils import can_dflash_slice_qkv_weight

def _stacked_ctx_kv_params(self) -> Optional[dict]:
    """Stack every layer's KV projection into one weight (exact: the input hidden is shared,
    so concatenating output columns is equivalent). Cached; None (per-layer fallback) when
    a QKV weight cannot be sliced (quantized) or layers disagree on norm epsilon / bias
    presence."""
    cached = getattr(self, "_stacked_ctx_kv_cache", False)
    if cached is not False:
        return cached # 命中缓存，直接返回

    weights, biases, k_norm_weights = [], [], []
    eps = None
    for layer in self.layers:
        attn = layer.self_attn
        can_slice, _ = can_dflash_slice_qkv_weight(attn.qkv_proj)
        # 必须可切片，且所有层 epsilon 相同
        if not can_slice or eps not in (None, attn.k_norm.variance_epsilon):
            self._stacked_ctx_kv_cache = None
            return None # 不符合条件，标记为 None 并快速返回
        eps = attn.k_norm.variance_epsilon
        kv_slice = slice(attn.q_size, attn.q_size + 2 * attn.kv_size)
        # 仅取 KV 部分（去掉 Q 部分）
        weights.append(attn.qkv_proj.weight[kv_slice])
        biases.append(
            attn.qkv_proj.bias[kv_slice] if attn.qkv_proj.bias is not None else None
        )
        k_norm_weights.append(attn.k_norm.weight)
```

```
    # 偏置存在性必须一致：要么全有，要么全无
    has_bias = [b is not None for b in biases]
    if any(has_bias) and not all(has_bias):
        self._stacked_ctx_kv_cache = None
        return None

    # 拼接并缓存
    self._stacked_ctx_kv_cache = {
        "weight": torch.cat(weights, dim=0),
        "bias": torch.cat(biases, dim=0) if all(has_bias) else None,
        "k_norm_weight": torch.stack(k_norm_weights, dim=0).float(),
        "eps": eps,
    }
    return self._stacked_ctx_kv_cache


def write_target_hidden_kv(self, *, target_hidden, pool, positions, cache_loc,
                           cache_loc_2d=None, commit_lens=None):
    ctx_hidden = self.project_target_hidden(target_hidden)
    stacked = self._stacked_ctx_kv_params() # 尝试使用堆叠路径
    if stacked is not None:
        # 堆叠路径：一次 GEMM + 批量 k-norm + RoPE
        k_all, v_all = self._project_ctx_kv_stacked(
            ctx_hidden=ctx_hidden, positions=positions, stacked=stacked
        )
    for i, layer in enumerate(self.layers):
        attn = layer.self_attn
        if stacked is not None:
            k = k_all[i]
            v = v_all[i]
        else:
            # 逐层回退路径
            k, v = attn.kv_proj_only(ctx_hidden)
            k = attn.apply_k_norm(k)
            k = attn.apply_k_rope(positions, k)
            k = k.view(-1, attn.num_kv_heads, attn.head_dim)
            v = v.view(-1, attn.num_kv_heads, attn.head_dim)
        # 写入 KV 缓存（略，与回退路径相同）
```

test/registered/spec/dspark/test_dspark_stacked_ctx_kv_parity.py

新增的数值一致性测试文件，是 PR 正确的关键保障，覆盖正常堆叠路径和三种回退条件。

```
def _per_layer_reference(model, ctx_hidden, positions):
    ks, vs = [], []
    for layer in model.layers:
        attn = layer.self_attn
        k, v = attn.kv_proj_only(ctx_hidden)
        k = attn.apply_k_norm(k)
        k = attn.apply_k_rope(positions, k)
```

```
            ks.append(k.view(-1, attn.num_kv_heads, attn.head_dim))
            vs.append(v.view(-1, attn.num_kv_heads, attn.head_dim))
        return ks, vs

    @unittest.skipUnless(torch.cuda.is_available(), "CUDA required")
    class TestDSparkStackedCtxKvParity(CustomTestCase):
        def setUp(self):
            super().setUp()
            set_global_server_args_for_scheduler(ServerArgs(model_path="dummy"))
            self.rope = get_rope(
                HEAD_DIM, rotary_dim=HEAD_DIM, max_position=4096,
                base=10000.0, is_neox_style=True
            ).to(DEVICE)

        def _check_parity(self, *, num_layers=4, tokens=5, has_bias=False, dtype):
            g = torch.Generator(device=DEVICE).manual_seed(0)
            model = _make_model(self.rope, num_layers, has_bias=has_bias, g=g)
            for layer in model.layers:
                attn = layer.self_attn
                attn.qkv_proj.weight = attn.qkv_proj.weight.to(dtype)
                if attn.qkv_proj.bias is not None:
                    attn.qkv_proj.bias = attn.qkv_proj.bias.to(dtype)
                attn.k_norm.to(dtype)
            ctx_hidden = torch.randn(tokens, HIDDEN, device=DEVICE, dtype=dtype, generator=g)
            positions = torch.arange(tokens, device=DEVICE)

            # 逐层参考路径
            ref_k, ref_v = _per_layer_reference(model, ctx_hidden, positions)
            # 堆叠路径
            stacked = model._stacked_ctx_kv_params()
            self.assertIsNotNone(stacked)
            k_all, v_all = model._project_ctx_kv_stacked(
                ctx_hidden=ctx_hidden, positions=positions, stacked=stacked
            )

            # 宽容度：fp16(5e-3), bf16(2e-2)
            rtol, atol = {torch.float16: (5e-3, 5e-3), torch.bfloat16: (2e-2, 2e-2)}[dtype]
            for i in range(num_layers):
                torch.testing.assert_close(k_all[i], ref_k[i], rtol=rtol, atol=atol)
                torch.testing.assert_close(v_all[i], ref_v[i], rtol=rtol, atol=atol)
```

评论区精华

本 PR 无实质 review 讨论，仅作者触发 CI 运行相关测试并确认通过。

- 暂无高价值评论线程

风险与影响

- 风险：风险点包括：（1）量化模型自动回退，性能收益受限，但不影响正确性。（2）缓存假设权重冻结，若运行时权重发生变化（如 LoRA 切换）将返回错误结果，但当前设计假设加载后冻结。（3）数值精度：fp32 k-norm 权重强制转换为 float32 可能导致微小差异，测试已覆盖 fp16/bf16 宽容度。（4）切片逻辑错误或层序不匹配可能产生错误输出，但 parity 测试直接覆盖了此场景。主要风险集中在 _project_ctx_kv_stacked 的实现正确性（未在提供材料中完全展示，需进一步审查）。
- 影响：影响范围仅限于 DSpark 草稿模型（DSparkDraftMixin 及其子类 DSparkDraftModel、Qwen3DSparkModel），不影响其他模型或通用推理路径。对用户而言，可降低推测解码延迟，提升吞吐。对系统无 API 或配置变更，不需要用户升级调整。对团队，新增方法封装良好，缓存自动失效机制简单，维护成本低。
- 风险标记：量化回退，数值精度敏感，核心路径变更

关联脉络

- PR #31985 [Perf] Fold dspark dense draft embedding into the draft graph via forward_embed: 同作者同模块优化，将密集草稿嵌入前移，共同改进 DSpark 推测解码效率
- PR #31981 [Perf] Skip page-table columns past kv length in DSA draft-extend metadata kernel: 同为推测解码性能优化，影响同一草稿场景