PR #26209 完整报告

sgl-project/sglang

Add FP4 Indexer for DeepSeek V4

合并时间：2026-06-02 15:14

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26209

执行摘要

- 一句话：为 DeepSeek V4 添加可选 FP4 索引器路径
- 推荐动作：建议精读此 PR，特别是 fp4_indexer.py 中的分组量化策略和 memory_pool.py 中按标志调整布局的设计。Review 中对 kernel 中间精度的讨论也值得学习。该 PR 体现了 SGLang 在模型推理中通过量化技术进行性能 - 精度权衡的典型实践。

功能与动机

DeepSeek V4 索引器默认使用 FP8 量化以节省内存和带宽，本 PR 进一步探索 FP4 量化，在可接受的精度损失（GSM8K 从 0.980 降至 0.965，GPQA pass@1 avg-of-16 从 0.835 变为 0.833）下获得更高推理吞吐。作者通过显式标志保持默认行为不变，提供性能与精度的权衡选择。

实现拆解

1. 新增 FP4 量化 Triton Kernel(python/sglang/srt/layers/attention/dsv4/fp4_indexer.py): 实现 _quantize_fp4_indexer_kernel 使用每 32 元素分组的 UE8M0 缩放因子和 E2M1 码本将 FP32/BF16 张量量化为 FP4，以及 _store_fp4_index_k_cache_kernel 将量化后的 K 数据和缩放因子存入页式缓存。
2. 扩展内存池适配 FP4 布局(python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py): DeepSeekV4IndexerPool 新增 get_bytes_per_token 方法根据 use_fp4_indexer 标志返回 index_head_dim // 2 + 4 字节（FP4 时）或原大小，从而调整缓存页大小。新增 set_index_fp4 和 set_index_k_fp4 方法委托给 store_fp4_index_k_cache。
3. 融合 Query 索引器 FP4 路径(python/sglang/jit_kernel/dsv4/elementwise.py 及 CUDA .cuh 文件): 新增 fused_q_indexer_rope_hadamard_fp4_quant 函数及其 JIT 编译模块，将 RoPE、Hadamard 变换和 FP4 量化融合为单一 CUDA kernel，减少中间数据移动。
4. 改造注意力前端分发(python/sglang/srt/layers/attention/dsv4/indexer.py, compressor.py, compressor_v2.py): 修改 forward_c4_indexer 和 _forward_prepare_multi_stream 等函数，当检测到 c4_indexer.use_fp4_indexer 时调用 compute_q_fp4 和 set_index_fp4。返回类型改为 Union[torch.Tensor, Tuple[torch.Tensor, torch.Tensor]] 以兼容 FP4 输出。
5. Benchmark 与测试(python/sglang/jit_kernel/benchmark/bench_dsv4_fp4_indexer.py, python/sglang/jit_kernel/tests/deepseek_v4/test_fp4_indexer.py): 提供 kernel 级

benchmark 对比 FP8/FP4 索引器；注册多个参数化测试验证量化、存储、融合 Q 路径的正确性。

关键文件：

- python/sglang/srt/layers/attention/dsv4/fp4_indexer.py（模块 索引量化；类别 source；类型 core-logic；符号 _select_group_value, _ceil_ue8m0_exp, _fp4_e2m1_code, _quantize_fp4_indexer_kernel）：新增 Triton FP4 量化和存储 kernel，是 FP4 索引器的核心技术实现
- python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 get_bytes_per_token, set_index_fp4, set_index_k_fp4）：修改 DeepSeekV4IndexerPool 以支持 FP4 内存布局，新增 get_bytes_per_token 和 set_index_fp4/set_index_k_fp4
- python/sglang/srt/layers/attention/dsv4/indexer.py（模块 索引器前端；类别 source；类型 dependency-wiring；符号 IndexerQuery, compute_q_fp4, forward_c4_indexer）：修改注意力前端分发逻辑以支持 FP4 索引器路径，引入 IndexerQuery 类型别名，新增 compute_q_fp4 方法
- python/sglang/jit_kernel/dsv4/elementwise.py（模块 JIT 核；类别 source；类型 core-logic；符号 _jit_main_q_indexer_rope_hadamard_fp4_quant_module, fused_q_indexer_rope_hadamard_fp4_quant）：新增 fused_q_indexer_rope_hadamard_fp4_quant 函数，将 RoPE、Hadamard 和 FP4 量化融合为单一 JIT kernel
- python/sglang/jit_kernel/tests/deepseek_v4/test_fp4_indexer.py（模块 测试套件；类别 test；类型 test-coverage；符号 _ceil_ue8m0_exp_ref, _fp4_e2m1_code_ref, _ref_quantize_fp4_indexer, _ref_store_fp4_index_cache）：新增全面测试，覆盖 FP4 量化、缓存存储、融合 Q 路径的正确性，参数化多种批量大小

关键符号：_quantize_fp4_indexer_kernel, _store_fp4_index_k_cache_kernel, quantize_fp4_indexer_tensor, store_fp4_index_k_cache, get_bytes_per_token, set_index_fp4, set_index_k_fp4, fused_q_indexer_rope_hadamard_fp4_quant, compute_q_fp4

关键源码片段

python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py

修改 DeepSeekV4IndexerPool 以支持 FP4 内存布局，新增 get_bytes_per_token 和 set_index_fp4/set_index_k_fp4

```
# deepseek_v4_memory_pool.py — FP4 索引器内存布局适配

class DeepSeekV4IndexerPool(KVCache):
    def __init__(self, ...):
        ...
        # 从全局配置读取 FP4 标志
        self.use_fp4_indexer = get_global_server_args().enable_deepseek_v4_fp4_indexer
        self._create_buffer()
```

```
def get_bytes_per_token(self) -> int:
    # FP4 下每个 token 只需 64 + 4 字节，FP8 下需要 128 + 4
    if self.use_fp4_indexer:
        return self.index_head_dim // 2 + 4
    return self.index_head_dim + 4

def _create_buffer(self):
    page_bytes = self.page_size * self.get_bytes_per_token()
    with self.memory_saver_adapter.region(GPU_MEMORY_TYPE_KV_CACHE):
        with ...:
            self.index_k_with_scale_buffer = [
                torch.zeros(
                    (self.size + self.page_size + 1) // self.page_size,
                    page_bytes,
                    dtype=self.index_k_with_scale_buffer_dtype,
                    device=self.device)
                for _ in range(self.layer_num)
            ]

def set_index_fp4(self, layer_id: int, loc: torch.Tensor, cache_k: torch.Tensor) -> None:
    from sglang.srt.layers.attention.dsv4.fp4_indexer import store_fp4_index_k_cache
    return store_fp4_index_k_cache(
        input=cache_k,
        cache=self.index_k_with_scale_buffer[layer_id - self.start_layer],
        loc=loc,
        page_size=self.page_size,
    )
```

评论区精华

Review 中主要讨论了五点：

- Hot path server_args 缓存：gemini-code-assist 建议避免在 forward 中重复调用 get_global_server_args()，作者改为在 attention backend 初始化时缓存标志。
- Fused Q FP4 kernel 需求：DarkSharpness 询问是否可以有融合 kernel，作者后续添加了 fused_q_indexer_rope_hadamard_fp4_quant。
- 测试位置整理：DarkSharpness 要求测试移至 JIT kernel 目录并遵循风格，作者执行了移动。
- 类型提示优化：gemini 建议使用 Union 替代 Any，作者引入 IndexerQuery 别名。
- CUDA kernel 中间精度：DarkSharpness 质疑为何通过 DType 往返转换，作者解释为了与未融合路径保持四舍五入一致，DarkSharpness 认可保留当前实现。
  - Hot path server_args 缓存 (performance): 作者改为在 attention backend 初始化时缓存 self.enable_deepseek_v4_fp4_indexer。
  - Fused Q FP4 kernel 设计 (design): 新增融合 kernel，避免 RoPE/Hadamard 中间数据移动。
  - 测试位置与风格整理 (testing): 作者将测试移至指定目录并精简。

- CUDA kernel 中间精度 (correctness): 保留当前实现，DarkSharpness 认可。
- 类型提示改进 (style): 作者引入 IndexerQuery 别名并应用于 prepare 路径返回类型。

风险与影响

- 风险:
  - 精度风险：FP4 量化在 GSM8K 上造成 1.5% 精度下降，在精度敏感场景需谨慎启用。
  - 架构限制：FP4 索引器路径依赖 DeepGEMM 的 fp8_fp4_paged_mqa_logits，目前仅支持 SM100 (Blackwell) 架构。在其他 GPU 上运行会报错或行为未定义。
  - 测试覆盖风险：虽然新增单元测试，但端到端测试仅限于长上下文固定输入的 benchmark，未覆盖短序列或不同 decode 模式。
  - 配置复杂度：新增 --enable-deepseek-v4-fp4-indexer 标志，与 --moe-runner-backend 等选项的交互未充分测试。
  - 影响：用户影响：FP4 索引器作为可选项，默认关闭。用户可通过命令行开启获得约 1.08x 吞吐提升，但需接受小幅精度回归。系统影响：修改了 DeepSeek V4 内存池的缓存页大小计算逻辑，但 FP8 路径完全不变。启动时会多编译一组 CUDA kernel，增加 JIT 编译时间但有缓存。团队维护：需要维护 FP4 和 FP8 两条索引器路径。融合 kernel 增加了代码库复杂度，但性能收益显著。
- 风险标记：精度下降风险，SM100 架构限制，核心索引路径变更

关联脉络

- PR #26931 [AMD] dpsk-v4 swa loc cache support: 同一模块 DeepSeekV4IndexerPool 的持续优化，memory_pool 文件变更重叠，均涉及 DeepSeek V4 索引器缓存改进。