PR #25144 完整报告

sgl-project/sglang

[NPU] Add Ascend NPU support for DeepSeek-V4

合并时间：2026-06-18 15:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25144

执行摘要

- 一句话：为 DeepSeek-V4 模型添加 Ascend NPU 完整推理支持
- 推荐动作：该 PR 值得仔细阅读，特别是对以下方面感兴趣的人：
  - 如何将复杂的注意力机制（混合 SWA + 压缩索引）移植到新硬件后端。
  - 如何通过工厂方法和钩子模式最小化对现有 CUDA 代码的侵入。
  - NPU 分页内存池与 CUDA 环缓冲在管理上的设计差异。 建议重点关注 ascend_dsv4_backend.py 中 Walsh-Hadamard 变换的实现、dsv4_memory_pool.py 中的 NPU 分页状态池大小计算、以及 dsv4_common_hooks.py 中的预写钩子模式。

功能与动机

PR 描述明确说明："Adds end-to-end Ascend NPU support for the DeepSeek-V4 architecture"。在 Ascend NPU 上运行 DeepSeek-V4 模型需要单独的底层实现，该 PR 填补了这一空白，使得 sglang 能够在华为昇腾设备上支持最新的 DeepSeek V4 系列模型。

实现拆解

1. 新增 NPU 专属核心文件：在 hardware_backend/npu/ 下创建注意力后端（ascend_dsv4_backend.py）、内存池（dsv4_memory_pool.py）、分页分配器（dsv4_allocator.py）、分配后钩子（dsv4_common_hooks.py）和请求映射池（dsv4_req_to_token_pool.py），完成 NPU 上 SWA + 压缩 KV 的完整生命周期管理。
2. 重构公共工厂方法：修改 mem_cache/deepseek_v4_memory_pool.py，将 KV 池、索引器池和状态池的构造提取为 _make_kv_pool、_make_indexer_pool、_make_attn_state_pool 等可重写方法，使 NPU 子类能无缝替换环缓冲为分页状态池（cache_mode=1）。
3. 模型与层适配：在 models/deepseek_v4.py 中为 NPU 添加分支，使用 torch_npu.npu_rms_norm 替代 triton RMSNorm，并集成自定义算子 torch.ops.custom.npu_hc_pre；在 layers/deepseek_v4_rope.py 中添加 NPU 兼容的 interleaved-RoPE 实现；使 layers/mhc.py 和 layers/attention/dsv4/compressor.py 中的 tilelang 导入变为可选，避免在 NPU 上加载失败。
4. 量化与 MoE 适配：在 layers/quantization/compressed_tensors/ 中新增 NPUCompressedTensorsW8A8Int8DynamicMoE 方案，将 W8A8 int8 MoE 映射到 NPU 内核；在 hardware_backend/npu/moe/topk.py 中添加 fused_hash_topk_npu 路由，通

过 torch.ops.custom.npu_moe_gating_top_k 加速 MoE 门控。

5. 调度与参数配置：在 managers/schedule_batch.py 中计算 DSV4 状态长度（_compute_dsv4_state_lens_{extend,decode}），在 arg_groups/deepseek_v4_hook.py 中根据设备强制选择 dsv4 注意力后端，并注册 dsv4 后端别名以解析到 NPU 子类。

关键文件：

- python/sglang/srt/hardware_backend/npu/attention/ascend_dsv4_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _walsh_hadamard_matrix, _apply_hadamard, _overlap_transform, CompressorAscendBackendMixin）：NPU 注意力后端的核心实现，包含 Walsh-Hadamard 变换、压缩元数据构建、NPU 特定前向路径，是整个 DSV4-NPU 推理的关键入口。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 NPUDeepSeekV4SingleKVPool, init, create_buffer, npu_state_pool_size）：NPU 特定 KV 池实现，替换 CUDA 的环缓冲为分页状态池（cache_mode=1），适配 Atlas A3 硬件约束。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_allocator.py（模块 分配器；类别 source；类型 core-logic；符号 get_last_loc, DSV4NPUTokenToKVPoolAllocator, init, mk）：NPU 分页分配器，在父类 SWA 分配基础上叠加 c4/c128 KV 和状态槽分配，返回六元组 DSV4OutCacheLoc。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_common_hooks.py（模块 公共钩子；类别 source；类型 core-logic；符号 maybe_write_dsv4_extend, maybe_write_dsv4_decode, _write_per_req, _write_state_tail_per_req）：分配后钩子，将 DSV4OutCacheLoc 写入每个请求的映射表，是连接分配器与注意力后端的桥梁。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_req_to_token_pool.py（模块 映射池；类别 source；类型 core-logic；符号 DSV4NPUReqToTokenPool, init, write_swa, write_c4）：扩展 ReqToTokenPool，为 SWA、c4、c128 及其状态添加辅助表，实现平台无关的映射。
- python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py（模块 缓存层；类别 source；类型 refactor；符号 _make_kv_pool, _make_indexer_pool, _state_pool_size, _make_attn_state_pool）：重构公共 KV 池工厂方法，为 NPU 子类提供可覆盖的构造点，是平台扩展的核心基础。
- python/sglang/srt/layers/deepseek_v4_rope.py（模块 位置编码；类别 source；类型 core-logic；符号 _yarn_get_mscale, _get_contig_freqs_real_imag, get_fused_compressor_rope_cos_sin, v4_rope_inplace_npu）：添加 NPU 兼容的 interleaved-RoPE 实现，包括连续 cos/sin 缓存和 inplace 旋转，并使 tilelang 导入可选。
- python/sglang/srt/layers/mhc.py（模块 MHC 层；类别 source；类型 dependency-wiring；符号 _TilelangMissing, getattr, _jit, _wrap）：使 tilelang 导入可选，并添加 NPU hc_pre 自定义算子路径，确保模块在 NPU 上正常加载。
- python/sglang/srt/models/deepseek_v4.py（模块 模型定义；类别 source；类型 core-logic）：在模型前向中添加 NPU 特定分支，使用 torch_npu.rms_norm、NPU RoPE 和自定义融合算子。

- python/sglang/srt/model_executor/forward_batch_info.py（模块 前向批信息；类别 source；类型 data-contract；符号 DSV4OutCacheLoc, DSV4StateLens）：新增 DSV4OutCacheLoc 和 DSV4StateLens 数据类，定义 NPU 分配结果的传递契约。

关键符号：_walsh_hadamard_matrix, _apply_hadamard, _overlap_transform, CompressorAscendBackendMixin._build_npu_compress_metadata, CompressorAscendBackendMixin._build_npu_compress_metadata_prefill, CompressorAscendBackendMixin._compute_compress_locs, CompressorAscendBackendMixin.forward_core_compressor, get_last_loc, DSV4NPUTokenToKVPoolAllocator.init, DSV4NPUTokenToKVPoolAllocator._compute_c_extend_counts, DSV4NPUTokenToKVPoolAllocator._alloc_state_extend, maybe_write_dsv4_extend, maybe_write_dsv4_decode, maybe_evict_dsv4_state, npu_state_pool_size, NPUDeepSeekV4SingleKVPool.create_buffer, DSV4NPUReqToTokenPool.write_swa, DSV4NPUReqToTokenPool.write_c4, DSV4NPUReqToTokenPool.write_c4_state, v4_rope_inplace_npu, fused_hash_topk_npu

关键源码片段

python/sglang/srt/hardware_backend/npu/dsv4/dsv4_memory_pool.py

NPU 特定 KV 池实现，替换 CUDA 的环缓冲为分页状态池（cache_mode=1），适配 Atlas A3 硬件约束。

```
def npu_state_pool_size(
    *,
    ratio: int,
    page_size: int,
    max_num_reqs: int,
) -> int:
    """NPU 分页状态池的槽位数量计算。

    公式：max(2, ceil(1.8 * ratio / page_size) + 1) * max_num_reqs * page_size
    基于稳态解码场景：每个请求只保留尾部滑动窗口长度的 state 槽，1.8 倍系数为页面边界留余量。

    结果以 token 为单位，匹配 SGLang 分配器 `PagedTokenToKVPoolAllocator(size, ...)` 的约定。
    """
    per_req_pages = max(2, (int(1.8 * ratio) + page_size - 1) // page_size + 1)
    # 缓存池大小 = 每请求页数 * 最大请求数 * 页面大小 (token 单位 )
    return per_req_pages * max_num_reqs * page_size


class NPUDeepSeekV4SingleKVPool(DeepSeekV4SingleKVPool):
    """NPU bf16 变体：使用全局 kernel_page_size 确保 c4/c128 池的页面宽度与 SWA 池一致。"""

    def __init__(self, *args, kernel_page_size: int, **kwargs):
        # 在 super().__init__ 之前设置 kernel_page_size，因为 _create_buffers() -> create_buffer()
        会读取它
        self.kernel_page_size = kernel_page_size
        super().__init__(*args, **kwargs)
```

```
    def create_buffer(self, *, num_pages: int):
        if self.store_dtype != torch.bfloat16:
            return super().create_buffer(num_pages=num_pages)
        kv_dim = self.qk_nope_head_dim + self.qk_rope_head_dim
        self.kv_cache_total_dim = kv_dim
        # PA_ND 布局：(num_pages, kernel_page_size, num_kv_heads=1, dim)
        npu_num_pages = (self.size + self.kernel_page_size + 1) // self.kernel_page_size
        return torch.zeros(
            npu_num_pages,
            self.kernel_page_size,
            1,
            kv_dim,
            dtype=torch.bfloat16,
            device=self.device,
        )
```

评论区精华

1. hc_pre 返回值不匹配（P1 问题）：chatgpt-codex-connector[bot] 指出 NPU 分支 hc_pre 返回 3 个值，但调用方 DeepSeekV4DecoderLayer.forward 解包 4 个，会导致 ValueError。Talantan1102 在 ab84745 中修复，添加了 False 作为第四个返回值。
2. Hadamard 变换归一化（P2 问题）：代码机器人指出 _apply_hadamard 直接使用 matmul(H) 而没有归一化，导致点积结果放大 sqrt(n)，使索引器 top-k 选择与 CUDA 路径不一致。Talantan1102 在 ab84745 中乘以 n**-0.5 修正。
3. 代码风格与导入位置：多位 reviewer（zhuyijie88, silencejade, sigama-w）要求将局部 import 移到文件顶部、移除不必要注释、简化日志；Talantan1102 在 89ef8d8 等提交中进行了调整。
4. tilelang 导入保护：mhc.py 和 deepseek_v4_rope.py 中使 tilelang 导入可选，silencejade 建议添加日志提示；Talantan1102 在 except 中增加了 logger.info。
5. 内存池修改范围：randgun 建议减少对 mem_cache/deepseek_v4_memory_pool.py 的修改，将 NPU 专用代码移到 dsv4_memory_pool.py；该建议被部分采纳，公共文件仍保留了工厂方法重构。

- hc_pre NPU 分支返回值不匹配 (correctness): Talantan1102 在 commit ab84745 中修复，在返回元组末尾添加 False。
- Hadamard 变换缺少归一化 (correctness): Talantan1102 在 commit ab84745 中乘以 n**-0.5 进行归一化。
- 导入位置与代码风格 (style): Talantan1102 在 commit 89ef8d8 中统一将 import 提升到文件顶部，并折叠冗长注释。
- tilelang 导入保护 (design): Talantan1102 在 except 中增加了 logger.info 提示信息。
- 公共内存池修改范围 (design): 部分采纳：公共文件保留了工厂方法的重构，但 NPU 特定实现全部移至 dsv4_memory_pool.py。

风险与影响

- 风险：
  1. 依赖自定义算子：NPU 路径依赖 torch.ops.custom.* 系列算子（如 npu_hc_pre、npu_sparse_attn_sharedkv、npu_moe_gating_top_k），这些算子仅在特定 CANN 版本（cann-8.5.0-a3）上可用。若部署环境不满足，启动时或运行时将崩溃。
1. 缺少单元测试覆盖：本次变更新增约 4K 行 NPU 代码，但未包含配套单元测试，仅通过 CI 集成测试验证，风险集中在边缘场景（如空 batch、prefix 为零、全 1 压缩率）。
2. 分页状态池兼容性：新引入的 NPU 分页状态池（cache_mode=1）与 CUDA 环缓冲（cache_mode=2）设计不同。若用户设置 SGLANG_DSV4_NPU_FUSED_COMPRESSOR=0 走 unfused 路径，translate_kv_loc_to_compress_state_loc 会直接报错，需要确保配置文件正确。
3. 分离式部署（disagg）不完全支持：dsv4_common_hooks.py 中的 TODO 明确指出分离式部署路径未调用预写钩子，会导致 c 页面泄漏，当前不推荐在分离式模式下使用 DSV4-NPU。 - 影响：用户影响：使 Ascend NPU 用户首次能在 sglang 上运行 DeepSeek-V4 全系列模型（Flash/Pro），推理过程自动使用 NPU 定制的 sparsed attention 和 fused compressor，获得性能收益。 系统影响：新增约 4.1K 行 NPU 专用代码，同时在公共路径中插入了少量平台无关的钩子（如 mem_cache/common.py 中的 DSV4 调度），对 CUDA 路径无影响。 团队影响：NPU 硬件后端模块得到显著扩展，为未来支持更多模型类型（如其他 NSA 架构）铺平了道路，但维护成本增加。
- 风险标记：依赖自定义算子，缺少测试覆盖，核心路径变更，分页状态池兼容性，分离式部署不完全支持

关联脉络

- PR #30111 [Fix] Fix DSA indexer fusion for NeoX RoPE: 同样涉及 DeepSeek 系列模型的注意力后端修复，展示了 DSA indexer 与不同 RoPE 方案的兼容性调整。
- PR #29959 [DSA][GLM5.2] Index Share for MHA: 为 MHA 添加 indexer 共享逻辑，与 DSV4-NPU 中压缩索引器（C4Indexer）的适配属于同一注意力基础设施演进。