PR #33676 完整报告

sgl-project/sglang

[NPU] Support DeepSeek-V4 DSpark and refactor DSV4 cache management

合并时间：2026-08-17 16:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33676

执行摘要

- 一句话：NPU 支持 DSV4 DSpark 推测解码并重构缓存所有权
- 推荐动作：值得精读，尤其关注三点设计决策：一是 C128 sidecar 所有权模型——只让“完整物理页”进入 Radix 树、部分尾页归请求所有，避免回退到更短前缀的匹配退化和页级引用计数复杂度；二是 cache_mode=2 显式状态块表对 GPU 与 A3 ABI 的适配（_build_explicit_state_block_table 的掩码列 clamp + dummy 正数偏移），这是平台差异的优雅收敛点；三是“NPU 专属逻辑下沉到 hardware_backend/npu 并用 guard 隔离”的分层策略。建议后续补上 C128 sidecar 的单元测试与 compact/ragged verify 路径。

功能与动机

PR body 明确动机："This PR adds NPU support for DeepSeek-V4 DSpark speculative decoding. It also consolidates the DSV4 NPU cache and memory-pool implementation and includes several small execution-path optimizations." 此前 DSV4 的 DSpark 推测解码只支持 CUDA，NPU 用户无法使用该能力；同时旧 NPU 缓存实现为融合压缩算子单独设计了一套 paged state pool（cache_mode=1），与 GPU 端 ring 所有权规则分叉，导致缓存分配、请求表、P/D 传输中存在重复与不一致的记账逻辑，需要收敛。

实现拆解

本 PR 的变更入口有两个：python/sglang/srt/models/deepseek_v4_dspark.py（决定 NPU 上能否加载 DSpark 模型并跑三段 draft 前向）与 python/sglang/srt/hardware_backend/npu/attention/ascend_dsv4_backend.py（决定压缩器 / 索引器元数据是否匹配 A3 算子 ABI）。实现可拆成 5 步：

1. DSpark 模型接入与 NPU 权重契约：deepseek_v4_dspark.py 新增 _remap_dspark_weight_name_npu，只接受 mtp.* 前缀权重并按 self_attn/ffn/markov_head 等规则重映射；NPU checkpoint 自带 QuaRot 对齐的 embed/LM head，故 uses_own_vocab_modules = is_npu()，attach_shared_modules 在 NPU 上不再共享 target 词表模块；shared_experts_fusion_disable_reason 在 NPU 上返回禁用原因（ModelSlim 权重映射不支持共享专家融合）。modelslim/modelslim.py 配套增加 W4A8/W8A8 的 NPU 兼容处理。
2. 压缩器后端切到 cache_mode=2 显式状态表：ascend_dsv4_backend.py 新增 _build_explicit_state_block_table，把 GPU 风格 state location 适配成 A3 的 table ABI：C4 状态经 full→SWA→state 两级翻译，C128 状态按 req_pool_idx + 绝对位置 寻址；

无效 / 历史 padding 位置统一写入 dummy_state_loc（正数、清零行），规避 A3 算子不认 -1 的问题。删除原 _overlap_transform 及其 cache_mode=1 的 state page table 清零逻辑，prefill metadata 中 seqused 显式计算、dsv4_max_input_capacity 按批内最大 chunk 长度构造。

3. 缓存与内存池所有权重构：这是 commit 0d5167e 的核心。dsv4_memory_pool.py 中 NPUCompressStatePool 从 paged 池退化为 GPU ring 状态池的薄适配层（继承分配 /ring 所有权 / 地址翻译，仅补 FP32 断言、3-D 视图和 dummy 行）；dsv4_allocator.py 删除 C4 KV 与两套状态分配器，C4 位置改由 _derive_c4_loc_from_full 从 full 槽位派生（% 4 == 3 判定完整组），C128 升级为独立物理页大小并带 c128_page_refcount；dsv4_req_to_token_pool.py 把原来的 req_to_token_swa/c4/c128/c4_state/c128_state 五张表收敛为一张 req_to_c128_sidecar，并支持 Radix 匹配结果经 Req.c128_prefix_page_ids 临时挂载后由 alloc 安装；dsv4_common_hooks.py 收窄为只写 C128 sidecar、只构造 C128 KV 的 PD payload，SWA/C4 状态复用公共传输路径。
4. 新增 Radix C128 sidecar 组件：全新 c128_sidecar_component.py 实现 C128SidecarComponent(TreeComponent)，有意识地把“完整 C128 物理页”作为 Radix 树上可匹配的最小单元、部分尾页仍归请求所有；插入时按 128 * page_size 组边界物化树节点并 _attach page，驱逐时维护设备释放与引用计数，且 C128 页不参与公开 token 驱逐计数，配合会话覆盖推进 / 回退与节点分裂时的 SWA rebuild 对齐。
5. 执行路径优化、图重放与 P/D 配套：deepseek_v4.py 新增 _forward_prepare_multi_stream_npu，把 KV 投影与 Q 投影分别放到独立 NPU 流上与索引器 / 压缩器重叠，并显式管理事件依赖；条件带 not is_extend_or_draft_extend_or_mixed()，只覆盖 decode/verify。dspark_worker_v2.py、dflash_info.py 适配 graph 重放与 DP-attention idle rank；新增 dflash_disaggregation.py 与 extra_ops_loader.py，后者通过 SGLANG_DSPARK_EXTRA_OPS_SO 环境变量加载并校验独立 .so 算子（npu_sparse_attn_sharedkv(_metadata)）。

测试与部署配套：test_npu_ascend_dsv4_backend.py 删除了 _overlap_transform 的 7 个单测（该函数被显式状态表取代）；未新增 NPU DSpark 端到端单测，验证依赖单节点 16 NPU 手动启动脚本与 GPQA-Diamond 精度（0.894）。临时外部算子依赖（Compressor、DSpark 稀疏注意力）通过 ASCEND_CUSTOM_OPP_PATH / SGLANG_DSPARK_EXTRA_OPS_SO 注入，待 sgl-kernel-npu#689/#699/#708 合入后移除。

关键文件：

- python/sglang/srt/hardware_backend/npu/dsv4/c128_sidecar_component.py（模块 Radix 缓存；类别 source；类型 core-logic；符号 C128SidecarComponent, _attach, commit_insert_component_data, finalize_match_result_in_cache）：本次缓存重构的核心新增：C128 sidecar 所有权模型，把完整 C128 物理页挂到 Radix 树节点，贯穿插入、分裂、驱逐、会话推进全生命周期。
- 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 _build_explicit_state_block_table, CompressorAscendBackendMixin, _build_npu_compress_metadata, _build_npu_compress_metadata_prefill）：压缩器 / 索引器后端从 cache_mode=1 paged 状态池切换到 cache_mode=2 显式状态块表，新增 A3 算子 ABI 适配，是本 PR 正

确性最敏感的文件。

- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_allocator.py（模块 分配器；类别 source；类型 core-logic；符号 _derive_c4_loc_from_full, retain_c128_pages, release_c128_pages, replace_req_c128_prefix）：分配器大幅简化：删除独立 C4 与状态分配器，C4 位置从 full 派生，C128 变为带引用计数的独立物理页分配。
- 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 NPUCompressStatePool, _replace_invalid_with_dummy, translate_from_swa_loc_to_state_loc, translate_from_req_position_to_state_loc）：NPUCompressStatePool 从 paged cache_mode=1 池改为 GPU ring 状态池薄适配层，C128 池引入独立物理页大小，是缓存重构的关键契约。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_common_hooks.py（模块 缓存钩子；类别 source；类型 core-logic；符号 maybe_write_dsv4_extend, dsv4_state_payloads, c128_kv_pages, write_dsv4_prealloc_tables）：公共缓存钩子从五张表写入收窄为只写 C128 sidecar，P/D payload 只保留 NPU 特有的 C128 KV 页，SWA/C4 复用公共路径。
- 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 write_c128, set_c128_prefix_pages, alloc, _init_dsv4_tables）：per-req 请求表由五张收敛为一张 req_to_c128_sidecar，并支持 Radix 匹配结果临时挂载后由 alloc 安装，是 sidecar 写入链路的落点。
- python/sglang/srt/hardware_backend/npu/extra_ops_loader.py（模块 算子加载；类别 source；类型 dependency-wiring；符号 OpLibSpec, TorchOpLoader, initialize, initialize_dspark_sparse_attn_ops）：新增外部算子库加载器，DSpark 稀疏注意力算子尚未进 sgl-kernel-npu 发布包，通过环境变量注入 .so，是部署门槛的直接体现。
- python/sglang/srt/models/deepseek_v4_dspark.py（模块 DSpark 模型；类别 source；类型 data-contract；符号 _remap_dspark_weight_name_npu, uses_own_vocab_modules, shared_experts_fusion_disable_reason）：NPU DSpark 模型权重契约的核心：ModelSlim checkpoint 的 NPU 权重名重映射、自带 embed/LM head、禁用共享专家融合。
- python/sglang/srt/models/deepseek_v4.py（模块 模型注意力；类别 source；类型 core-logic；符号 _forward_prepare_multi_stream_npu）：新增 NPU 多流 forward_prepare，KV/Q 分离到独立流与索引器 / 压缩器重叠，评审要求只覆盖 decode/verify。
- python/sglang/srt/speculative/dflash_disaggregation.py（模块 分离式调度；类别 source；类型 core-logic；符号 build_dflash_family_disagg_draft_input）：新增 DFlash 族模型在 P/D 分离下的 draft 输入构建，支持 overlap 模式的 future publish/stash，是 DSpark 调度配套。
- test/registered/unit/npu/attention/test_npu_ascend_dsv4_backend.py（模块 注意力测试；类别 test；类型 test-coverage；符号 TestOverlapTransform）：删除了被 _build_explicit_state_block_table 取代的 _overlap_transform 的全部 7 个单测，未补等价覆盖，是测试配套缩减的注意点。

关键符号：C128SidecarComponent.commit_insert_component_data, C128SidecarComponent._attach, C128SidecarComponent.finalize_match_result_in_cache, C128SidecarComponent.evict_component, _build_explicit_state_block_table, CompressorAscendBackendMixin._build_npu_compress_metadata, CompressorAscendBackendMixin._build_npu_compress_metadata_prefill, DSV4NPUTokenToKVPoolAllocator._derive_c4_loc_from_full, DSV4NPUTokenToKVPoolAllocator.retain_c128_pages, NPUCompressStatePool._replace_invalid_with_dummy, NPUCompressStatePool.translate_from_swa_loc_to_state_loc, DeepseekV4ForCausalLMDSpark._remap_dspark_weight_name_npu, DeepSeekV4Attention._forward_prepare_multi_stream_npu, DSV4ReqToTokenTablesMixin.write_c128, DSV4ReqToTokenTablesMixin.set_c128_prefix_pages, TorchOpLoader.initialize, build_dflash_family_disagg_draft_input

关键源码片段

python/sglang/srt/hardware_backend/npu/dsv4/c128_sidecar_component.py

本次缓存重构的核心新增：C128 sidecar 所有权模型，把完整 C128 物理页挂到 Radix 树节点，贯穿插入、分裂、驱逐、会话推进全生命周期。

```
class C128SidecarComponent(TreeComponent):
    """C128 sidecar 所有权模型：只把完整的物理 C128 page 暴露给 Radix 树。

    部分尾页仍然由请求自身持有，避免把未闭合的压缩组写进共享前缀。
    """
    component_type = ComponentType.C128

    @property
    def allocator(self):
        # DSV4 NPU 分配器持有 C128 page 引用计数，归还由它统一处理。
        return self.cache.token_to_kv_pool_allocator

    def _attach(self, node: UnifiedTreeNode, pages: torch.Tensor) -> None:
        if pages.numel() == 0:
            return
        ct = self.component_type
        cd = node.component_data[ct]
        assert cd.value is None
        value = pages.clone() # 克隆后再挂树，避免后续写入污染共享前缀
        self.tree_core.set_component_device_value(node.id, ct, value)
        self.allocator.retain_c128_pages(value) # 增加 page 引用计数，防提前回收

    def commit_insert_component_data(
        self,
        node: UnifiedTreeNode,
        is_new_leaf: bool,
```

```
        params: InsertParams,
        result: InsertResult,
        cache_actions: list[CacheAction | ComponentAction],
    ) -> None:
        if not is_new_leaf:
            return
        assert params.key is not None
        assert params.c128_value is not None

        # Full/SWA 可能先把新后缀表示成一条长叶子；这里把每个完整的
        # C128 组边界都物化成树节点并挂上对应 page，让后续分支总能匹配到
        # 最近的完整 C128 page 前缀，而不是回退到更短的旧 Radix 节点。
        group_tokens = 128 * self.allocator.c128_attn_allocator.page_size
        first_boundary = (result.prefix_len // group_tokens + 1) * group_tokens
        for boundary in range(first_boundary, len(params.key) + 1, group_tokens):
            boundary_node = self._ensure_boundary_node(node, boundary, cache_actions)
            page_index = boundary // group_tokens - 1
            self._attach(boundary_node, params.c128_value[page_index : page_index + 1])

    def evict_component(
        self,
        node: UnifiedTreeNode,
        device_frees: dict[ComponentType, list[torch.Tensor]],
        host_frees: dict[ComponentType, list[torch.Tensor]],
        target: EvictLayer = EvictLayer.DEVICE,
    ) -> tuple[int, int]:
        cd = node.component_data[self.component_type]
        if EvictLayer.DEVICE in target and cd.value is not None:
            device_frees[self.component_type].append(cd.value)
            self.tree_core.component_evictable_size_[self.component_type] -= len(cd.value)
            cd.value = None
        # C128 页是 Full token 的辅助数据，不参与公开 token 驱逐计数。
        return 0, 0
```

python/sglang/srt/hardware_backend/npu/attention/ascend_dsv4_backend.py

压缩器 / 索引器后端从 cache_mode=1 paged 状态池切换到 cache_mode=2 显式状态块表，新增 A3 算子 ABI 适配，是本 PR 正确性最敏感的文件。

```
  def _build_explicit_state_block_table(
      *,
      compress_ratio: int,
      coff: int,
      state_pool,
      token_to_kv_pool,
      req_to_token: torch.Tensor,
      req_pool_indices: torch.Tensor,
      start_pos: torch.Tensor,
      cu_seqlens: torch.Tensor,
```

```
    seqused: torch.Tensor,
    max_input_capacity: int,
) -> torch.Tensor:
    """把 GPU 风格的 state location 适配成 A3 cache_mode=2 的 table ABI。

    cache_mode=2 需要一张显式的 state_block_table：历史列 + 当前输入列，
    列宽固定为 coff * ratio + max_input_capacity；无效/越界位置写入
    dummy_state_loc（正数、清零行），因为 A3 算子只接受无符号偏移。
    """
    req_pool_indices = req_pool_indices.to(torch.int64)
    capacities = cu_seqlens[1:] - cu_seqlens[:-1]
    history_size = coff * compress_ratio
    width = history_size + max_input_capacity
    columns = torch.arange(width, dtype=torch.int64, device=req_to_token.device)
    positions = start_pos[:, None] - history_size + columns
    within_capacity = columns[None, :] < history_size + capacities[:, None]
    valid = (seqused[:, None] > 0) & within_capacity & (positions >= 0)

    if compress_ratio == 4:
        # C4 状态跟随 SWA 物理页：先 full→SWA 再 SWA→state 两级翻译。
        # 掩码列仍会在 torch.where 之前被索引，所以先 clamp 防越界。
        safe_positions = positions.clamp(0, req_to_token.shape[1] - 1)
        full_locs = req_to_token[req_pool_indices[:, None], safe_positions]
        swa_locs = token_to_kv_pool.translate_loc_from_full_to_swa(full_locs)
        state_locs = state_pool.translate_from_swa_loc_to_state_loc(swa_locs)
    else:
        # C128 状态按 req_pool_idx + 绝对位置寻址，走独立侧车映射。
        state_locs = state_pool.translate_from_req_position_to_state_loc(
            req_pool_indices[:, None], positions
        )

    return torch.where(
        valid,
        state_locs.to(torch.int32),
        state_pool.dummy_state_loc,
    ).contiguous()
```

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

NPUCompressStatePool 从 paged cache_mode=1 池改为 GPU ring 状态池薄适配层，C128 池引入独立物理页大小，是缓存重构的关键契约。

```
class NPUCompressStatePool(CompressStatePool):
    """A3 上的薄适配层：直接复用 GPU 版 ring 状态池的分配与所有权规则。

    旧实现为 NPU 单独做了一套 paged state pool（cache_mode=1，block 0 作
    skip-sentinel）；新实现改走 cache_mode=2 显式位置表，因此状态缓冲的
    分配、ring 所有权和地址翻译全部继承自 CompressStatePool。NPU 侧只需：
      1. 提供连续 3-D 视图；
      2. 强制 FP32 契约（A3 custom.compressor 只接受 FP32 state_cache）；
```

```
    3. 把非法位置替换为已清零的"正数"dummy 行——显式模式下位置 0 合法，
       不能再用 -1 表示无效（A3 算子消费无符号偏移）。
    """

    def __init__(self, *, size, overlap, head_dim, dtype, device,
                 enable_memory_saver, ratio, ring_size, swa_page_size):
        assert ratio in (4, 128), f"NPUCompressStatePool 只支持 ratio 4/128，收到 {ratio}"
        assert dtype == torch.float32, f"Atlas A3 custom.compressor 需要 FP32 state_
        cache，收到 {dtype}"
        assert ring_size > 0, f"ring_size 必须为正，收到 {ring_size}"

        super().__init__(
            size=size, ring_size=ring_size, overlap=overlap, head_dim=head_dim,
            dtype=dtype, device=device, enable_memory_saver=enable_memory_saver,
            ratio=ratio, online=False, swa_page_size=swa_page_size,
            state_cache_page_size=ring_size,
        )
        # 末行预先清零，作为"无效"位置的落点；A3 算子吃无符号偏移。
        self.dummy_state_loc = self._size - 1

        if ratio == 128:
            # 共享池只会初始化 dummy 行；冷启动的 C128 请求 bank 在首次
            # 部分使用前需要把每一行都初始化好。
            self.kv_score_buffer.clear()

    def _replace_invalid_with_dummy(self, state_loc: torch.Tensor) -> torch.Tensor:
        return torch.where(
            state_loc < 0,
            torch.full_like(state_loc, self.dummy_state_loc),
            state_loc,
        )
```

评论区精华

评审共 5 条 comment，集中在三处：

1. verify 宽度读取来源（AndyLi429）：在 ascend_dsv4_backend.py 的 _build_npu_compress_metadata 上提问 use spec_info.draft_token_num? ，针对原实现用 self.speculative_num_draft_tokens 的问题。head 已采纳为 int(forward_batch.spec_info.draft_token_num)，确保图重放时 verify 元数据与当前 speculative forward 的真实 draft 数量一致。
2. 池类型判断方式（randgun）：在 disaggregation/utils.py 第 1168 行建议 maybe use isinstance(token_to_kv_pool, DSV4NPUTokenToKVPool) is better?，属于可读性建议；材料中未见明确落地，需保留为待核对项。
3. 多流不要覆盖 extend（randgun）：在 deepseek_v4.py 第 1390 行要求多流启用条件 change to not extend。head 最终条件已带上 not forward_batch.forward_mode.is_extend_or_draft_extend_or_mixed()，即 NPU 多流只作用于 decode/verify。

- verify 宽度改读 spec_info.draft_token_num (correctness): head 已改为 int(forward_batch.spec_info.draft_token_num)，并配套抽出 n_draft 变量，评审意见被采纳。
- DSV4 池类型判断建议用 isinstance (design): 材料中未见明确落地记录，建议保留为待核对项。
- NPU 多流条件不要覆盖 extend (performance): head 最终条件已加上 not forward_batch.forward_mode.is_extend_or_draft_extend_or_mixed()，多流只作用于 decode/verify，反馈已落实。

风险与影响

- 风险：技术风险集中在四个层面：

1. 核心路径大重构：DSV4 NPU 缓存从五表 + paged 状态池改为 sidecar + ring 所有权，C4 位置从“独立分配”变为“从 full 派生”（依赖 % 4 == 3 的对齐假设）。任何 full 槽位分配顺序或段边界变化都会静默影响 C4 定位，回归风险高，dsv4_allocator.py / ascend_dsv4_backend.py 需要重点回归。
2. 外部算子临时依赖：Compressor（torch.ops.custom.compressor）、npu_sparse_attn_sharedkv(_metadata) 等算子尚未进 sgl-kernel-npu 发布包，必须通过 SGLANG_DSPARK_EXTRA_OPS_SO 与 ASCEND_CUSTOM_OPP_PATH 手动注入；extra_ops_loader.py 只校验了算子存在与 Python ABI，未校验底层 CANN 版本兼容性，部署失败面较宽。
3. 验证覆盖有限：PR 明确 NPU compact/ragged verify 未启用，仅验证 static verify；同时删除了 _overlap_transform 单测且未补等价覆盖，C128 sidecar 的 Radix 插入 / 驱逐 / 会话推进也没有对应单测，图捕获路径下的正确性主要靠 16 NPU 手工验证。
4. CUDA 隔离依赖 guard：NPU 专属逻辑散落在 deepseek_v4.py、deepseek_v4_dspark.py 等共享模型文件中，靠 _is_npu 分支隔离；后续若有人调整共享路径（如 DSV4StateLens 结构体删除、forward_batch_info.py 中 -40 行），可能遗漏 NPU 分支的联动更新。 - 影响：影响范围：对 NPU 用户而言是能力级提升——ModelSlim 量化的 DeepSeek-V4 搭配 DSpark 推测解码首次可在 Ascend A3 上运行，GPQA-Diamond 0.894 且打通 DP attention、DeepEP、P/D 分离；对 DSV4 缓存管理是一次跨模块重构，涉及分配器、内存池、请求映射池、Radix 缓存、公共钩子与 P/D 传输，改变了 NPU 与 CUDA 在压缩器状态所有权上的分叉程度（收敛到 GPU ring 规则），后续两平台维护成本降低但合入期回归面大。团队影响：36 个 commit、4 位作者（2044145178、randgun、Talantan1102、JiaruiChang5268）协作，含两个大 squash（perf 集成与缓存重构），主线合并频繁，属于典型的并行分支长期演进合入。 - 风险标记：核心路径变更，外部算子临时依赖，验证覆盖受限，测试配套减少，CUDA 路径隔离依赖，多平台并行演进

关联脉络

- PR #33480 [AMD] Support prefill context parallel two batch overlap for DeepSeek V4: 同为 DSV4 上的多流 / 批次重叠性能路径，本 PR 在 NPU 侧补 _forward_prepare_multi_stream_npu 与索引器多流，两平台形成对照实现。

- PR #35059 [Spec] Resolve shared-read ends from the backend declaration alone: speculative 调度与共享读栅栏的后端声明重构，与本 PR 的 draft worker、verify 元数据改动同属推测解码链路演进。
- PR #34277 [DSV4] Emit TMA-aligned UE8M0 scales for FP8 einsum: DSV4 量化内核与 FP8 路径优化，与本 PR 的 ModelSlim 量化加载及 DSV4 后端同属 DSV4 特性线。