PR #28980 完整报告

sgl-project/sglang

[NPU] Support DeepSeek V4 Flash MTP on Ascend

合并时间：2026-06-30 16:22

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28980

执行摘要

- 一句话：NPU 上支持 DeepSeek V4 Flash MTP 推理
- 推荐动作：建议对所有文件进行仔细审查，尤其是 ascend_dsv4_backend.py 的 multi-step draft backend 实现和 dsv4_allocator.py 的状态备份机制。这些设计可复用于其他平台的多步推测解码场景。PR 虽大但模块化较好，值得精读以理解 NPU 上 DSV4 speculative decoding 的全链路。

功能与动机

Enable DeepSeek V4 Flash ModelSlim NEXTN/MTP inference on Ascend NPU. 需要为 NPU 适配 DSV4 压缩 KV 的分配、verify 阶段的元数据构建以及 multi-step draft backend 的初始化和执行路径。

实现拆解

1. Ascend DSV4 后端增强 (ascend_dsv4_backend.py)：添加 _build_npu_compress_metadata_verify 方法，在 target-verify 模式生成 c4/c128 压缩位置元数据；新增 DeepseekV4AscendMultiStepDraftBackend 类，提供 multi-step draft 的前向 / 存储钩子。
2. NPU DSV4 分配器扩展 (dsv4_allocator.py)：新增 alloc_paged_token_slots_reserve_extend 函数，在预留 extend 时计算 DSV4 state lens 并调用通用分配器；提供 backup_state/restore_state 用于 MTP 场景的 state 临时恢复。
3. 公共钩子增强 (dsv4_common_hooks.py)：maybe_write_dsv4_extend 增加显式偏移参数，支持预留场景；新增 maybe_build_dsv4_verify_bundle 函数，从 per-req 表格构建 verify 阶段的缓存位置束。
4. 推测解码路由 (eagle_utils.py, eagle_info_v2.py)：在 prepare_for_decode 和 prepare_for_verify 中根据设备类型分发到 NPU 分配器，并插入 verify bundle 构建逻辑。
5. NEXTN 模型适配 (deepseek_v4_nextn.py)：对 ModelSlim 检查点使用 mtp.0 前缀替代默认 decoder 以匹配权重命名约定。
6. 其他清理：移除了部分不再需要的调试日志、环境变量和临时权重映射代码。注意本次未包含独立的 NPU 测试文件，仅依赖 CPU 端 verify-bundle 测试。

关键文件：

- python/sglang/srt/hardware_backend/npu/attention/ascend_dsv4_backend.py（模块 NPU 后端；类别 source；类型 dependency-wiring；符号 _build_npu_compress_metadata_verify, _fill_verify_positions_cmp_padding, _fill_verify_positions_cmp_padding_one, update_verify_buffers_to_fill_after_draft）：核心后端文件：实现了压缩元数据 verify 生成和 multi-step draft backend，包含新类 DeepseekV4AscendMultiStepDraftBackend。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_allocator.py（模块 分配器；类别 source；类型 core-logic；符号 alloc_paged_token_slots_extend_npu, alloc_paged_token_slots_reserve_extend, compute_dsv4_state_lens_reserve, backup_state）：核心分配器文件：新增预留分配入口和状态备份 / 恢复函数，支撑 MTP 场景的 cache 管理。
- 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_build_dsv4_verify_bundle, flatten_interval）：公共钩子文件：增强 maybe_write_dsv4_extend 支持自定义 offset，新增 maybe_build_dsv4_verify_bundle 构建 verify 束。
- python/sglang/srt/speculative/eagle_utils.py（模块 推测解码；类别 source；类型 dependency-wiring）：推测解码工具：根据设备类型路由到 NPU 分配器，并在 prepare_for_verify 中插入 verify bundle 构建。
- python/sglang/srt/models/deepseek_v4_nextn.py（模块 NEXTN 模型；类别 source；类型 data-contract）：NEXTN 模型适配：对 ModelSlim 检查点使用 'mtp.0' 前缀，以匹配权重命名。

关键符号：_build_npu_compress_metadata_verify, alloc_paged_token_slots_reserve_extend, maybe_build_dsv4_verify_bundle, backup_state, restore_state, compute_dsv4_state_lens_reserve, update_verify_buffers_to_fill_after_draft, _fill_verify_positions_cmp_padding

关键源码片段

python/sglang/srt/hardware_backend/npu/dsv4/dsv4_allocator.py

核心分配器文件：新增预留分配入口和状态备份 / 恢复函数，支撑 MTP 场景的 cache 管理。

```
def alloc_paged_token_slots_reserve_extend(
    tree_cache,
    prefix_lens: torch.Tensor,
    prefix_lens_cpu: torch.Tensor,
    seq_lens: torch.Tensor,
    seq_lens_cpu: torch.Tensor,
    last_loc: torch.Tensor,
    extend_num_tokens: int,
    *,
    req_pool_indices: Optional[torch.Tensor] = None,
    dsv4_state_lens: Optional[DSV4StateLens] = None,
    batch=None,
):
```

```
    """
    在预留 extend 时分配 slot，并更新 DSV4 per-request 表格。
    如果 dsv4_state_lens 未传入，则调用 allocator.compute_dsv4_state_lens_reserve
    从 batch.reqs 计算。然后调用通用 alloc_paged_token_slots_extend 进行实际分配，
    最后通过 maybe_write_dsv4_extend 将新分配的 slot 写入 per-req 表格。
    """
    # 若未提供 state lens 则从 batch 计算
    if dsv4_state_lens is None and batch is not None:
        allocator = batch.token_to_kv_pool_allocator
        dsv4_state_lens = (
            allocator.compute_dsv4_state_lens_reserve(
                batch.reqs, prefix_lens_cpu, seq_lens_cpu
            )
            if hasattr(allocator, "compute_dsv4_state_lens_reserve")
            else None
        )
    # 调用通用分配函数
    out_cache_loc = alloc_paged_token_slots_extend(
        tree_cache,
        prefix_lens, prefix_lens_cpu,
        seq_lens, seq_lens_cpu,
        last_loc, extend_num_tokens,
        req_pool_indices=req_pool_indices,
        dsv4_state_lens=dsv4_state_lens,
        batch=batch,
    )
    # 写入 per-req 表格（显式设置 offset 为 prefix 位置）
    if batch is not None:
        maybe_write_dsv4_extend(
            batch,
            batch.req_pool_indices_cpu,
            prefix_lens_cpu, seq_lens_cpu,
            c4_state_alloc_offsets=prefix_lens_cpu,
            c128_state_alloc_offsets=prefix_lens_cpu,
        )
    return out_cache_loc
```

评论区精华

Review 讨论主要围绕以下几点：

- dsv4_state_lens 不应暴露在公共函数中：gjsheu 指出 dsv4_state_lens 参数不应出现在公共接口中，应移至后端内部实现。
- 现有路径是否需要额外的 DeepSeek V4 判断：gjsheu 和 randgun 质疑在 ascend_backend.py 和 draft_utils.py 中新增的 is_deepseek_v4 判断是否必要，后调整分支结构。
- 预填充阶段的压缩逻辑：gjsheu 评论预填充阶段应使用原生算子，不应额外压缩 metadata 生成，相关代码被移除。

- 代码归属与社区兼容：多个 reviewer 要求将非 NPU 通用的更改移至 NPU 后端目录，最终大部分改动限定在 npu 目录。
- 安全性警告：chatgpt-codex-connector 指出添加 NEXTN 可能导致 speculative_eagle_topk 为 None 时断言失败。
  - dsv4_state_lens 不应出现在公共函数中 (design): 相关参数被移除或改为可选，并通过 allocator.compute_dsv4_state_lens_reserve 内部计算。
  - ascend_backend.py 添加 DeepSeek V4 判断是否必要 (design): 经确认，DSV4 走专用后端，不应影响通用路径，相关判断被移除或调整分支顺序。
  - 预填充阶段压缩逻辑应移除 (design): 开发者（qybnb）回复 'no works, to be deleted'，相关代码被标记删除。
  - topk.py 的 NPU 特殊处理可能破坏社区分支 (correctness): 改成先检查 is_npu 并 return，再处理原本的 masked 逻辑。

风险与影响

- 风险：
  1. 回归风险：allocator 和 backend 的修改通过 is_deepseek_v4 隔离，但仍存在影响非 DSV4 路径的风险。
  2. 性能风险：verify 阶段压缩元数据生成可能引入额外开销，但通过预分配和一屏式计算最小化。
  3. 兼容性风险：dsv4_common_hooks.py 的 TODO 明确标注 disagg 路径绕过钩子导致 compressed 页面泄漏，未修复。
  4. 缺少测试覆盖：PR 没有新增 NPU 端到端测试或单元测试，仅 CPU 少量验证，质量风险高。
  5. 跨平台影响：eagle_utils.py 和 deepseek_v4_nextn.py 的修改通过设备类型分支，reviewer 确保了对 CUDA 路径的最小侵入。 - 影响：正面影响：首次在 Ascend NPU 上支撑 DeepSeek V4 Flash 模型的 speculative decoding（MTP），显著提升 NPU 上的推理吞吐。影响范围：仅限 NPU 平台且模型为 DeepSeek V4 系列。团队影响：NPU 后端维护者需了解新的 multi-step draft backend 和 verify bundle 机制。用户影响：NPU 用户可启用 --speculative-algorithm NEXTN 获得加速，但需要相应的 ModelSlim 检查点。 - 风险标记：缺少测试覆盖，核心路径变更（DSV4 分配器），与 disagg 路径兼容性未解决

关联脉络

- PR #29420 [AMD][DSV4] Remove per-batch D2H syncs in MTP to avoid bubbles between 2 batches: 同为 DSV4 和 speculative decoding 优化，但针对 AMD 平台；本 PR 借鉴了类似的 multi-step draft 设计思路。