PR #35676 完整报告

sgl-project/sglang

[NPU] DeepSeek-V4 adapt sgl-kernel-npu ops (compressor/sparse-attn/sparse-attn-metadata)

合并时间：2026-08-25 15:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35676

执行摘要

- 一句话：NPU 上 DSV4 稀疏注意力适配新算子，metadata 移至 host 计算
- 推荐动作：值得精读：host 侧 metadata 计算与 CUDA graph 捕获 / 回放下 CPU 镜像的构建策略是核心设计决策，对理解多硬件（NPU/XPU/AMD）后端适配很有价值；randgun 对简化条件的坚持与 vstone-w 对算子的澄清也值得留意。建议阅读时重点关注 CPU 镜像一致性与 sgl-kernel-npu 版本绑定，若要在生产环境启用，建议先补一次 PR 前后性能对比。

功能与动机

PR body 明确指出两个动机：一是为 NPU 平台适配 DSV4 稀疏注意力，对接 sgl-kernel-npu 提供的三个算子（compressor PR#689、sparse_attn_sharedkv PR#708、sparse_attn_sharedkv_metadata_host PR#699）；二是把 sparse_attn_sharedkv_metadata 从设备侧搬到 host 侧计算——原先设备张量需要 D2H 同步，'eliminating the D2H sync that drains/stalls the stream and hurts overlapped scheduling'。此外，统一 op 命名空间让 draft 与 normal 路径使用相同 kernel，减少双路径维护成本与行为分叉风险。

实现拆解

1. 统一 DSV4 算子命名空间：ascend_dsv4_backend.py 的 forward_compress 中 torch.ops.custom.compressor 改为 torch.ops.npu.compressor，稀疏注意力相关调用一并替换为 torch.ops.npu.*；dspark_worker_v2.py 的 init_attention_backends 删除 NPU 分支下的 initialize_dspark_sparse_attn_ops 调用，使 draft 与 normal 路径共用同一套 kernel。
2. host 侧 metadata 计算：ascend_backend.py 的 ForwardMetadata 新增 actual_seq_lengths_q_pa_cpu 字段；ascend_dsv4_backend.py 在 init_forward_metadata 中按 prefill（cumsum 前缀和）、decode（0..B 等差）、draft_extend（按 n_draft 步长）三种模式构建 CPU 镜像；_init_dsv4_graph_metadata 在 CUDA graph 捕获期一次构建镜像；_refresh_graph_seq_metadata 新增 else 分支用 final_seq_lens_cpu 维护 seq_lens_cpu_int 兜底，同时修复 _use_host_sparse_metadata 开关关闭时的 fallback；_kernel_metadata_from_parts 的 common 参数中移除 cu_seqlens_q、seqused_kv，改由 host op 直接读 CPU 镜像。
3. 移除冗余算子加载：extra_ops_loader.py 删除 initialize_dspark_sparse_attn_ops 及 _C_ascend 下 npu_sparse_attn_sharedkv_metadata 的必需校验，并为 TorchOpLoader

补充详细 docstring，说明环境变量定位、依赖预加载等用法；dspark_worker_v2.py 不再单独触发加载。

4. 分配路径与配置读取修复：allocation.py 的 get_last_loc 将安全分支条件从 _is_hip 扩展为 (_is_hip or _is_npu) and uses_triton_dispatch，NPU DSV4 分配路径改走 int32-safe 的 get_last_loc_triton_safe；dsv4_memory_pool.py 适配 compressor 的 FP32 state-cache 约束；xpu_backend.py 将确定性推理开关的读取从 model_runner.server_args 改为 get_exec().deterministic，符合 config bag 读取规范（配合 CI 的 config ratchet 检查）。
5. 测试与精度验证：test_npu_ascend_backend.py 的 test_field_names 补充 actual_seq_lengths_q_pa_cpu 断言；PR body 给出 aime26 精度 DSPARK ON/OFF 均为 0.9667、serving benchmark 总吞吐 26686.16 tok/s、TPOT 中位数 29.19 ms，但没有提供 PR 前的基线对比。

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 forward_compress, init_forward_metadata, _kernel_metadata_from_parts, _refresh_graph_seq_metadata）：DSV4 NPU 注意力后端核心，完成 host 侧 metadata 计算（CPU 镜像维护）与 torch.ops.npu 命名空间统一，改动量最大、风险最集中。
- python/sglang/srt/hardware_backend/npu/extra_ops_loader.py（模块 算子加载；类别 source；类型 core-logic；符号 TorchOpLoader, initialize_dspark_sparse_attn_ops）：删除 DSpark 专属算子加载函数 initialize_dspark_sparse_attn_ops 及 _C_ascend 依赖校验，是统一命名空间的关键配套，也是 review 重点讨论文件。
- python/sglang/srt/mem_cache/allocation.py（模块 显存分配；类别 source；类型 core-logic；符号 get_last_loc）：将 NPU 纳入 get_last_loc_triton_safe 安全路径，修复 NPU DSV4 分配路径可能出现的 int32->int64 越界问题，影响所有 NPU 平台。
- python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py（模块 投机解码；类别 source；类型 dependency-wiring；符号 init_attention_backends）：移除 draft worker 重复的算子加载调用，normal/draft 路径共用统一命名空间，是命名空间统一在投机解码路径上的落点。
- python/sglang/srt/hardware_backend/npu/attention/ascend_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 ForwardMetadata）：ForwardMetadata 新增 actual_seq_lengths_q_pa_cpu 字段，是 host metadata 计算的数据结构基础。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_memory_pool.py（模块 显存池；类别 source；类型 core-logic）：适配 compressor 对 FP32 state-cache 的内存池约束，是融合 compressor 在 NPU 上正确运行的前提。
- python/sglang/srt/layers/attention/xpu_backend.py（模块 XPU 后端；类别 source；类型 configuration；符号 num_splits）：顺带修复 enable_deterministic_inference 的读取方式，从 server_args 改为 config bag，配合 CI config ratchet 检查。
- test/registered/unit/npu/attention/test_npu_ascend_backend.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_field_names）：为新增的 actual_seq_lengths_q_pa_cpu 字段补充字段名断言，是唯一的测试配套改动。

关键符号：initialize_dspark_sparse_attn_ops, get_last_loc, get_last_loc_triton_safe, forward_compress, init_forward_metadata, _kernel_metadata_from_parts, _refresh_graph_seq_metadata, _init_dsv4_graph_metadata, init_attention_backends

关键源码片段

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

DSV4 NPU 注意力后端核心，完成 host 侧 metadata 计算（CPU 镜像维护）与 torch.ops.npu 命名空间统一，改动量最大、风险最集中。

```
# 关键实现单元：为 host metadata 算子维护 CPU int32 镜像输入。
# 背景：NPU 的 torch.ops.npu.sparse_attn_sharedkv_metadata_host 只接受
# CPU int32 张量；若在运行时对 device 张量做 .cpu()，会触发 D2H 同步，
# drain 当前 stream 并打断 overlapped 调度。因此 forward metadata 里
# 新增 actual_seq_lengths_q_pa_cpu 镜像，按 forward 模式构建一次后，
# 直接供 _kernel_metadata_from_parts 组装 kernel 参数时取用。

# prefill：每个请求的 KV 边界 = cumsum(seq_lens)，前面补 0 得到 cu_seqlens
fm.actual_seq_lengths_q_pa_cpu = torch.cat(
    [torch.zeros(1, dtype=torch.int32), torch.cumsum(seq_lens_cpu, dim=0).int()],
    dim=0,
)

# decode：每个请求只产出 1 个 token，边界即 0..B 的等差数列
fm.actual_seq_lengths_q_pa_cpu = torch.arange(0, B + 1, dtype=torch.int32)

# graph capture：q_pa 按 graph shape 恒定、replay 从不改写，所以捕获期
# 一次构建 CPU 镜像即可；kv 侧长度则从 CPU 源直接 clamp 生成，同样避开 D2H
metadata.actual_seq_lengths_q_pa_cpu = torch.arange(
    0, bs * tokens_per_req + tokens_per_req, tokens_per_req, dtype=torch.int32
)
fm.seq_lens_cpu_int = ctx.final_seq_lens_cpu[:ctx.bs].int().clamp(min=1)
```

python/sglang/srt/mem_cache/allocation.py

将 NPU 纳入 get_last_loc_triton_safe 安全路径，修复 NPU DSV4 分配路径可能出现的 int32->int64 越界问题，影响所有 NPU 平台。

```
def get_last_loc(req_to_token, req_pool_indices_tensor, prefix_lens_tensor):
    prefill_backend, decode_backend = attention_backends()
    uses_triton_dispatch = prefill_backend not in ("ascend", "torch_native") and \
        decode_backend not in ("ascend", "torch_native")

    # HIP 与 NPU DSV4：旧 get_last_loc_triton 内核会写出 int32 -> int64
    # 混合宽度存储，Triton 在 HIP 上会错误编译出越界的 last_loc，NPU DSV4
    # 的等价分配路径也可能 fault。统一走 int32-safe 变体，在 launch 之后
    # 才提升为 int64；其余硬件 / 后端保持原分发逻辑。
    if (_is_hip or _is_npu) and uses_triton_dispatch:
```

```
        return get_last_loc_triton_safe(
            req_to_token, req_pool_indices_tensor, prefix_lens_tensor
        )

    impl = get_last_loc_triton if uses_triton_dispatch else get_last_loc_torch
    return impl(req_to_token, req_pool_indices_tensor, prefix_lens_tensor)
```

评论区精华

randgun 在 extra_ops_loader.py 上建议直接删除 initialize_dspark_sparse_attn_ops 及其调用、用注释说明 extra_ops_loader 的用法（已采纳，函数与调用点均已删除，TorchOpLoader 补上详细 docstring）；在 allocation.py 上建议去掉 uses_safe_last_loc 中间变量，直接写作if(_is_hipor_is_npu)anduses_triton_dispatch（head版本已按此落地）。AndyLi429 询问 host 新算子与 npu_sparse_attn_sharedkv_metadata 是否同一个、是否重命名，vstone-w 澄清两者不同、不能互换；对于测试字段新增，vstone-w 解释 actual_seq_lengths_q_pa_cpu 是 host metadata op 的入参（只接受 host tensor）。ping1jing2 表示 LGTM 但建议补充 PR 前后性能对比，该建议在合入时未落实。

- 删除 initialize_dspark_sparse_attn_ops (design): 已删除该函数及其调用点，并扩充 TorchOpLoader 的 docstring 说明用法
- get_last_loc 安全路径的条件简化 (design): 采纳建议，head 版本直接使用该条件，去掉中间变量
- host metadata 算子与 npu_sparse_attn_sharedkv_metadata 的关系 (question): vstone-w: not same，this is used only for dspark, and can not be replaced by each other
- 新增 actual_seq_lengths_q_pa_cpu 字段断言的必要性 (testing): vstone-w: param for sparse_attn_sharedkv_metadata_host, which only accept host tensor

风险与影响

- 风险： 1) 核心路径一致性风险：ascend_dsv4_backend.py 把 metadata 输入改为 host 侧镜像，graph 捕获期一次构建、replay 期复用，若设备侧序列长度变化而镜像未同步，稀疏注意力 kernel 会读到错误元数据；seq_lens_cpu_int 的 clamp(min=1) 兜底逻辑需要额外关注。 2) 版本绑定风险：统一到 torch.ops.npu.* 依赖 sgl-kernel-npu 对应算子版本（PR#689/#699/#708），且 extra_ops_loader.py 移除了 metadata op 的加载后校验，旧 .so 环境下错误会延迟到运行时才暴露。 3) 跨平台辐射：allocation.py 的条件扩展为 _is_hip or _is_npu，影响所有 NPU 平台的 last_loc 计算，不限于 DSV4，虽理论上更安全但覆盖范围大于本 PR 目标。 4) 配置读取改动：xpu_backend.py 改为 get_exec().deterministic，依赖运行时上下文已初始化，若在构造期不可用可能导致 XPU 启动回归。 5) 测试覆盖有限：单测仅补一个字段断言，缺少对 host metadata op 的定向单测；性能数据没有 PR 前基线，收益难以量化。
  - 影响：影响面集中在 NPU + DSV4/DSpark 路径：对部署 DeepSeek-V4（含 DSpark 投机解码）的 NPU 用户，D2H 同步消除可改善 overlapped 调度与吞吐，精度保持不变（aime26 0.9667）。统一命名空间降低了 normal/draft 双路径的 kernel 维护成本，但要求 sgl-kernel-npu 升级到对应算子版本，形成版本绑定。allocation.py 的改动会辐射

所有 NPU triton 分配路径（行为更安全）。xpu_backend.py 的改动只影响 XPU 上的确定性推理开关读取方式。团队侧，NPU 后端与 sgl-kernel-npu 的协作模式更清晰，DSV4 多硬件适配脉络（XPU/NPU/AMD）进一步收敛。

- 风险标记：核心路径变更（DSV4 稀疏注意力后端），依赖 sgl-kernel-npu 新算子版本，缺少 PR 前性能基线，单测覆盖有限

关联脉络

- PR #32166 [XPU] Use SYCL kernels for DeepSeek V4 MHC on XPU: 同为 DeepSeek-V4 在不同硬件后端的 kernel 适配，NPU 与 XPU 两条适配路径可对照
- PR #35672 [AMD] Enable draft_extend CUDA graph for HIP DSA backend: 同为 speculative draft 路径的硬件适配，与本 PR 的 DSpark draft worker 改动相关
- PR #36300 config: the model-config cache keys on the path the record carried: 本 PR 末尾 commit 对 xpu_backend 的 config bag 读取修复同属 server_args/config 读取规范问题
- PR #36240 [CI] Stop the config ratchets re-parsing the package on every scan: config ratchet 系列检查要求从 config bag 读配置，xpu_backend 的改动正是为此类检查通过而做的配套