PR #31948 完整报告

sgl-project/sglang

[NPU] Enable automatic ascend_attn selection for vision attention and graph runners

合并时间：2026-08-02 15:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31948

执行摘要

- 一句话：NPU 多模态注意力默认选 ascend_attn，图执行同步生效
- 推荐动作：值得精读。该 PR 是理解 SGLang 多模态后端选择机制的良好示例：它展现了“单一解析点 + 结果传播”如何避免组件间重新读取原始配置的脆弱性，并演示了硬件优化实现必须与安全默认（mask 回退）配套的设计权衡。关注 NPU 或多模态推理的工程师建议仔细阅读 vision.py 中 backend 解析与 VisionAscendAttention 的 mask 回退逻辑。

功能与动机

PR body 指出，Ascend NPU 此前在 VisionAttention._determine_attention_backend() 中没有平台特定默认值，用户必须显式传 --mm-attention-backend ascend_attn，否则多模态模型会静默回退到 torch.nn.functional.scaled_dot_product_attention（SDPA）。即使补上默认值，ViT 图运行器仍会再次读取 ServerArgs.mm_attention_backend，用户省略参数时该值为 None，导致“eager 已选中 ascend_attn 但图捕获却报 unsupported backend”的不一致。PR 期望的行为是：显式 server 参数 > 模型 / 构造参数 > 平台默认，且 eager 与图运行器使用同一个已解析的 backend。

实现拆解

实现分为四步：

1. 补 NPU 平台默认并固化 backend 名称（python/sglang/srt/layers/attention/vision.py） - 在 _determine_attention_backend() 中新增 elif _is_npu: backend = "ascend_attn"，使 NPU 与其他平台一样拥有平台默认值。 - 在 VisionAttention.__init__ 中解析完成后新增 self.qkv_backend_name: str = qkv_backend，把最终生效的 backend 名称固化在 attention 实例上，供外部图运行器读取。 - 显式 server 参数与构造参数优先级保持不变，且不修改全局 ServerArgs.mm_attention_backend。
2. 图运行器改用已解析的 backend（vit_cuda_graph_runner.py、internvl_vit_cuda_graph_runner.py、vit_npu_graph_runner.py） - 三个运行器在 __init__ 中通过 getattr(self._attn, "qkv_backend_name", None) 缓存 backend。 - 原先在 _create_graph / _warmup_once / _capture_graph 中反复调用 get_mm().mm_attention_backend 的逻辑改为使用 self._attn_backend，保证 eager 与图路径一致。 - 不支持的 backend 报错信息统一带上实际 backend 名称，便于定位。 - NPU 运行器还删除了对 get_mm 的 import，避免对全局参数的隐式依赖。

3. Ascend 注意力增加 mask 兼容回退（vision.py） - 基于 review 意见，VisionAscendAttention 增加 sdpa_fallback = VisionSdpaAttention(**kwargs)，forward 中一旦收到显式 attention_mask 立即转交 SDPA，因为 Ascend 融合算子 npu_fused_infer_attention_score 不支持 additive mask，而 Mllama 和 CLIP 文本路径必须保留 mask 语义。
4. 测试配套 - 新增 test/registered/unit/layers/attention/test_vision_backend_selection.py：覆盖 NPU 平台默认选择优先级（server/ 构造参数 / 平台默认），以及 causal/padding 两种 mask 下回退 SDPA、unmasked 输入保持融合路径的回归用例。 - 扩展 test/registered/unit/multimodal/test_vit_cuda_graph_runner.py：验证 ViTCudaGraphRunner 与 InternViTCudaGraphRunner 构造后正确缓存 qkv_backend_name。 - 测试同时注册了 CPU CI 与 NPU CI 套件；真机在 Ascend 910B2C 上验证了 eager 与 ViT NPU 图捕获 / 回放全流程。

关键文件：

- python/sglang/srt/layers/attention/vision.py（模块 视觉注意力；类别 source；类型 core-logic；符号 VisionAttention._determine_attention_backend, VisionAttention.init, VisionAscendAttention.init, VisionAscendAttention.forward）：核心变更：为 NPU 增加 ascend_attn 平台默认、固化 qkv_backend_name、并为 Ascend 融合注意力增加 mask 时的 SDPA 回退，是全局解析与兼容性策略的落点。
- python/sglang/srt/multimodal/vit_cuda_graph_runner.py（模块 图运行器；类别 source；类型 dependency-wiring；符号 ViTCudaGraphRunner.init, ViTCudaGraphRunner._create_graph）：通用 ViT CUDA graph runner 改为消费 attention 模块已解析的 qkv_backend_name，确保用户省略 CLI 参数时图捕获仍能与 eager 使用同一后端。
- python/sglang/srt/multimodal/internvl_vit_cuda_graph_runner.py（模块 图运行器；类别 source；类型 dependency-wiring；符号 InternViTCudaGraphRunner.init, InternViTCudaGraphRunner._warmup_once, InternViTCudaGraphRunner._capture_graph）：InternVL 专用 CUDA graph runner 同步切换到 resolved backend，warmup 与 capture 保持一致，并改进错误信息。
- python/sglang/srt/hardware_backend/npu/graph_runner/vit_npu_graph_runner.py（模块 图运行器；类别 source；类型 dependency-wiring；符号 ViTNpuGraphRunner._create_graph）：NPU ViT graph runner 继承通用 runner 的 _attn_backend 缓存逻辑，使未传 CLI 参数时 NPU 图捕获能正确识别 ascend_attn。
- test/registered/unit/layers/attention/test_vision_backend_selection.py（模块 后端选择测试；类别 test；类型 test-coverage；符号 npu_platform, test_npu_backend_selection_priority, test_ascend_attention_masked_inputs_fall_back_to_sdpa, test_ascend_attention_unmasked_inputs_keep_fused_path）：新增的回归测试覆盖 NPU backend 选择优先级与 Ascend 注意力的 mask 回退 / 融合路径，是本次行为变更的安全网。
- test/registered/unit/multimodal/test_vit_cuda_graph_runner.py（模块 图运行器测试；类别 test；类型 test-coverage；符号 test_vit_graph_runner_caches_resolved_backend_name, test_internvl_graph_runner_caches_resolved_backend_name, Block, forward）：

补充图运行器从 attention 模块缓存 backend 名称的回归测试，防止未来重读全局参数的回归。

关键符号：VisionAttention._determine_attention_backend, VisionAttention.init, VisionAscendAttention.init, VisionAscendAttention.forward, ViTCudaGraphRunner.init, ViTCudaGraphRunner._create_graph, InternViTCudaGraphRunner.init, InternViTCudaGraphRunner._warmup_once, InternViTCudaGraphRunner._capture_graph, ViTNpuGraphRunner._create_graph

关键源码片段

python/sglang/srt/layers/attention/vision.py

核心变更：为 NPU 增加 ascend_attn 平台默认、固化 qkv_backend_name、并为 Ascend 融合注意力增加 mask 时的 SDPA 回退，是全局解析与兼容性策略的落点。

```
# python/sglang/srt/layers/attention/vision.py
# ==== VisionAttention：backend 解析与固化 ====
# 优先级：ServerArgs 显式参数 > 模型构造参数 > 平台默认。
# NPU 平台此前没有默认值，用户必须手动传 --mm-attention-backend ascend_attn；
# 现在补上平台默认，并把解析结果写入 qkv_backend_name 供 graph runner 直接读取。
class VisionAttention(nn.Module):
    def __init__(self, ..., qkv_backend=None):
        ...
        _passed_backend = qkv_backend
        qkv_backend = self._determine_attention_backend(_passed_backend)
        # 固化生效 backend 名称，避免 graph runner 重读原始 ServerArgs
        self.qkv_backend_name: str = qkv_backend
        ...

    def _determine_attention_backend(self, passed_backend):
        override_backend = get_mm().mm_attention_backend
        if override_backend is not None:
            return override_backend # 1. server 参数优先
        if passed_backend is not None:
            return passed_backend # 2. 构造参数其次
        if is_cuda():
            # CUDA 按算力选 fa3 / fa4 / triton_attn（分支省略）
            ...
        if _is_npu:
            return "ascend_attn" # 3. NPU 平台默认（本 PR 新增）
        # 其他平台（MUSA/ROCm/XPU/CPU）分支省略，最后回落 sdpa
        ...
        return "sdpa"

# ==== VisionAscendAttention：mask 兼容回退 ====
# Ascend 融合算子 npu_fused_infer_attention_score 不支持 additive mask，
# 因此收到显式 attention_mask 时转交 SDPA，保留 MIlama 与 CLIP 的 mask 语义。
class VisionAscendAttention(nn.Module):
    def __init__(self, **kwargs):
```

```
        if not _is_npu:
            raise Exception("VisionAscendAttention is only available for ascend npu")
        super().__init__()
        self.sdpa_fallback = VisionSdpaAttention(**kwargs)

    def forward(self, q, k, v, cu_seqlens, bsz, seq_len,
                softmax_scale=None, forward_metadata=None,
                attention_mask=None, **kwargs):
        if attention_mask is not None:
            return self.sdpa_fallback(
                q=q, k=k, v=v,
                cu_seqlens=cu_seqlens, bsz=bsz, seq_len=seq_len,
                attention_mask=attention_mask,
                forward_metadata=forward_metadata, **kwargs,
            )
        # 无 mask 时继续走下方 npu_fused_infer_attention_score 融合路径
        ...
```

python/sglang/srt/multimodal/vit_cuda_graph_runner.py

通用 ViT CUDA graph runner 改为消费 attention 模块已解析的 qkv_backend_name，确保用户省略 CLI 参数时图捕获仍能与 eager 使用同一后端。

```
# python/sglang/srt/multimodal/vit_cuda_graph_runner.py
# 此前 _create_graph 每次重新读取 get_mm().mm_attention_backend,
# 用户省略参数时得到 None，导致 eager 已选好平台默认而图捕获仍报错。
# 现在构造时直接缓存 attention 模块解析后的 backend 名称。
class ViTCudaGraphRunner:
    def __init__(self, vit: nn.Module) -> None:
        ...
        first_blk = vit.blocks[0]
        self._attn: Optional[VisionAttention] = getattr(first_blk, "attn", None)
        self._attn_backend: Optional[str] = getattr(
            self._attn, "qkv_backend_name", None
        )

    def _create_graph(self, graph_key, ...):
        graph = torch.cuda.CUDAGraph()
        ...
        # 使用解析后的 backend，而不是重读全局 ServerArgs
        backend = self._attn_backend
        ...
        if backend == "triton_attn":
            cu_seq_len_ws = [cu_seqlens_now, cu_seqlens_kk_now, max_len]
        elif backend == "fa3":
            cu_seq_len_ws = [cu_seqlens_now, max_len]
        else:
            raise RuntimeError(
                f"ViT CUDA graph does not support attention backend: {backend}"
            )
```

评论区精华

核心争议集中在 NPU 平台默认对 masked 调用者的影响：

> JustinTong0323（评论于 vision.py:1190）： 无条件增加 NPU 默认会导致所有 masked VisionAttention 调用者丢失 mask：VisionAscendAttention.forward() 忽略 attention_mask，而 Mllama 传入 aspect-ratio/padding mask、CLIP text 传入 causal mask，SDPA 之前会正确执行。要求先补 masked-attention 回归测试，或在 Ascend 后端实现 mask 支持。
>
> ltaodream（回复）： 已在提交 ed19574dc 中修复：VisionAscendAttention 收到显式 attention_mask 时回退到 SDPA，保留 Mllama 和 CLIP 的 mask 语义；未掩码输入继续走融核路径；新增 causal 与 padding 两种 mask 的回归测试，并在 Ascend 910B2C 上以 FP16/BF16 验证了 masked 与 unmasked 两条路径。

最终 JustinTong0323 给出 APPROVED。该讨论体现了“平台默认必须是安全的默认”这一设计原则，也说明硬件优化实现的能力边界（不支持 mask）需要显式兼容而不是静默忽略。

- NPU 平台默认会破坏 masked VisionAttention 调用者的 mask 语义 (correctness)：ltaodream 在提交 ed19574 中为 VisionAscendAttention 增加 sdpa_fallback，forward 收到 attention_mask 时转交 VisionSdpaAttention；新增 causal/padding 两种 mask 的回归测试，确保融合算子不被调用，并在 Ascend 910B2C 上验证。

风险与影响

- 风险：主要风险点：

1. NPU 用户默认行为变更：此前未显式指定 --mm-attention-backend 的 NPU 用户会从 SDPA 切到 ascend_attn。虽然 ascnd 融合算子更快，但数值行为可能与 SDPA 存在细微差异，且新增的 mask 回退依赖 attention_mask 参数被正确透传；如果某条调用链没有把 mask 传到 VisionAscendAttention.forward，可能静默丢失 mask 语义。
2. 图运行器契约变化：三个图运行器现在依赖 VisionAttention.qkv_backend_name 属性。若某模型的视觉注意力类不是 VisionAttention（未设置该属性），getattr 会返回 None，图捕获将报 unsupported backend。虽然这类模型此前也无法正确进入图捕获，但报错信息变化需要关注。
3. NPU 图路径的假设：ViTNpuGraphRunner._create_graph 仍只支持 ascend_attn，且依赖 cu_seq_lens 传递模式；若未来 NPU 后端增加其他实现，需同步扩展。
4. 验证范围：单元测试通过 monkeypatch 模拟 NPU 平台，真机验证仅在 910B2C 单型号上完成，未覆盖其他 Ascend 型号（如 910B、310P）以及更复杂的多 batch/ 可变分辨率组合。 - 影响：用户影响：Ascend NPU 上使用常见 VisionAttention 路径的多模态模型（如 Qwen-VL 系列）不再需要手动指定 --mm-attention-backend ascend_attn，开箱即用优化实现；显式传参的用户行为完全不变。

系统影响：eager 与 ViT 图执行路径首次在 backend 选择上达成一致，消除了“eager 可用但图捕获失败”的不一致状态；NPU ViT 图捕获可自动启用，首个请求图捕获约 6.14 秒，后续同形状请求回放约 0.17 秒。

团队影响：为后续新增平台默认后端提供了可参考的范式（解析结果固化到 qkv_backend_name，图运行器直接消费），同时 review 中暴露的 mask 兼容问题促使 Ascend 后端补上了必要的 SDPA 回退，提升了 NPU 多模态路径的健壮性。

- 风险标记：NPU 用户默认行为变更，mask 回退路径依赖，图运行器契约变化，真机验证范围有限

关联脉络

- PR #33168 Fix the chunked-prefix-cache gate writing config the backends never read: 与本 PR 同类问题：后端组件读取未解析的原始配置而非实际生效配置。本 PR 将 ViT graph runner 从重读 ServerArgs 改为消费 attention 模块解析后的 qkv_backend_name，与 33168 的修复思路一致。
- PR #33127 [Fix] Bound FULL_MASK verify-mask reuse by the captured max_bs: 同为图执行与 mask 正确性相关的修复，关注图捕获时 mask/backend 状态与运行期的一致性，可互为参照。
- PR #31221 [AMD] Derive AITER verify tokens-per-req from input shape: 同类平台后端修复：强调后端参数应当由实际输入或已解析状态推导，而不是依赖外部全局参数，与本 PR 消除 get_mm() 重读的模式一致。