PR #21197 完整报告

sgl-project/sglang

[NPU]adaptation to support deterministic inference

合并时间：2026-06-09 09:22

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/21197

执行摘要

- 一句话：NPU 后端确定性推理适配
- 推荐动作：值得精读，特别是 npu_batch_invariant_ops.py 和 batch_invariant_ops.py 中的条件注册逻辑，展示了如何在 SGLang 框架中为不同硬件后端提供操作符替换。采样器中的种子化改造也是一个可重用的模式。

功能与动机

根据 PR 描述：'We need to enable deterministic inference on the Ascend NPU to ensure consistent LLM outputs across runs when the --enable-deterministic-inference parameter is set, when graph mode is disabled.' 当前 NPU 后端在非图模式下无法保证同一输入输出一致，需要进行操作符级别的替换。

实现拆解

1. 新增 NPU batch-invariant 操作符层：创建 python/sglang/srt/hardware_backend/npu/batch_invariant_ops/npu_batch_invariant_ops.py，包装来自 batch_invariant_ops 开源库的 NPU 专用 batch-invariant 操作符，包括 npu_mm, npu_matmul, npu_mean, npu_log_softmax, npu_fused_infer_attention_score, npu_add_rms_norm。其中 npu_add_rms_norm 因底层算子不保证不变性，拆分为 add 和 npu_rms_norm 两步完成。
2. 修改 batch-invariant 调度入口：在 python/sglang/srt/batch_invariant_ops/batch_invariant_ops.py 的 enable_batch_invariant_mode 函数中，按平台分支处理：若为 NPU，则使用上述新的 NPU 操作符注册到 aten Library 的 IMPL 调度表，而非 CUDA 版本的实现。同时将 torch.ops.npu.npu_fused_infer_attention_score 和 torch_npu.npu_add_rms_norm 替换为 batch-invariant 版本。
3. 采样阶段确定性改造：在 python/sglang/srt/layers/sampler.py 中，将 Ascend 后端路径上的 torch.multinomial 替换为带种子的 multinomial_with_seed 调用；并在 top_k_top_p_min_p_sampling_from_logits_ascend 中同样引入种子分支。同时向相关函数传递 positions 用于种子生成。
4. 服务器参数调整：在 python/sglang/srt/server_args.py 中，将 "ascend" 加入 DETERMINISTIC_ATTENTION_BACKEND_CHOICES 和 RADIX_SUPPORTED_DETERMINISTIC_ATTENTION_BACKEND，使得 NPU 后端在确定性推理时可启用 radix cache。同时修改 _handle_deterministic_inference 中的采样后端设置逻辑：如果后端是 ascend，则保持为 ascend，否则强制为 pytorch。

5. 设备分发补全：在 python/sglang/srt/utils/common.py 中，向 get_dispatch_device_backend 添加 NPU 分支，返回 "NPU" 作为 dispatch key，确保后续操作符分发正确。
6. 其他清理：在 ascend_backend.py 中移除一个不正确的注释（“FIA supports multi-bs in the current version of CANN”），避免混淆。

关键文件：

- python/sglang/srt/hardware_backend/npu/batch_invariant_ops/npu_batch_invariant_ops.py（模块 NPU 后端；类别 infra；类型 infrastructure；符号 npu_mm_batch_invariant, npu_matmul_batch_invariant, npu_mean_batch_invariant, npu_log_softmax_batch_invariant）：新增文件，封装 NPU 专用的 batch-invariant 操作符，是确定性推理的核心基础。
- python/sglang/srt/layers/sampler.py（模块 采样器；类别 source；类型 core-logic）：确定性采样逻辑修改，将 Ascend 路径上的 torch.multinomial 替换为带种子的 multinomial_with_seed。
- python/sglang/srt/server_args.py（模块 服务配置；类别 source；类型 core-logic）：将 ascend 加入确定性注意力后端列表和 radix cache 支持列表，并在确定性推理时保留 ascend 采样后端。
- python/sglang/srt/batch_invariant_ops/batch_invariant_ops.py（模块 BatchInvariant；类别 infra；类型 infrastructure）：确定性调度入口，根据平台分支注册操作符，NPU 分支加载专用操作符。
- python/sglang/srt/utils/common.py（模块 工具函数；类别 source；类型 core-logic）：在 get_dispatch_device_backend 中添加 NPU 分支，使 CUDA/XPU 之外的设备能正确分发。
- python/sglang/srt/hardware_backend/npu/attention/ascend_backend.py（模块 NPU 注意力；类别 source；类型 core-logic）：移除误导性注释，代码无其他变更。

关键符号：_sample_from_logits, top_k_top_p_min_p_sampling_from_logits_ascend, _forward_ascend_backend, enable_batch_invariant_mode, _handle_deterministic_inference, get_dispatch_device_backend, npu_mm_batch_invariant, npu_mean_batch_invariant, npu_add_rms_norm_batch_invariant, npu_fused_infer_attention_score_batch_invariant

关键源码片段

python/sglang/srt/hardware_backend/npu/batch_invariant_ops/npu_batch_invariant_ops.py

新增文件，封装 NPU 专用的 batch-invariant 操作符，是确定性推理的核心基础。

```
# Adapted from https://github.com/thinking-machines-lab/batch_invariant_ops/blob/main/batch_invariant_ops/batch_invariant_ops.py
import batch_invariant_ops # noqa: F401
import torch
import torch_npu
```

```
def npu_mm_batch_invariant(a, b):
    """batch-invariant 矩阵乘法包装"""
    return torch.ops.batch_invariant_ops.npu_mm_batch_invariant(a, b)


def npu_matmul_batch_invariant(a, b):
    return torch.ops.batch_invariant_ops.npu_matmul_batch_invariant(a, b)


def npu_mean_batch_invariant(input, dim, keepdim=False, dtype=None):
    """对带 dim 参数求均值，内部根据 dim 数量选择不同实现：单 dim 用自定义 kernel，多 dim 用
    sum/除法"""
    assert dtype is None or dtype == torch.float32, f"unsupported dtype: {dtype}"
    if len(dim) == 1:
        return torch.ops.batch_invariant_ops.npu_reduce_mean_batch_invariant(input, dim[0],
        keepdim=keepdim)
    else:
        assert input.dtype in {torch.float16, torch.bfloat16, torch.float32}, "only float types
        supported"
        n_elems = 1
        for d in dim:
            n_elems *= input.shape[d]
        return torch.sum(input, dim=dim, keepdim=keepdim, dtype=torch.float32) / n_elems


def npu_log_softmax_batch_invariant(input, dim, _half_to_float):
    assert not _half_to_float, "not implemented" # _half_to_float 分支暂不支持
    return torch.ops.batch_invariant_ops.npu_log_softmax_batch_invariant(input, dim=dim)


def npu_fused_infer_attention_score_batch_invariant(*args, **kwargs):
    """NPU 融合注意力分数计算的 batch-invariant 版本"""
    return torch.ops.batch_invariant_ops.npu_fused_infer_attention_score_batch_invariant(*args,
    **kwargs)


def npu_add_rms_norm_batch_invariant(x, residual, weight, eps):
    """
    AclnnAddRmsNorm 不能保证 batch invariant,
    因此拆分为 add 和 rms_norm 两步：
    x_ = x + residual
    residual_ = x_  # 此处 residual_ 被赋值为 x_，但后续未使用（兼容原接口）
    x_, _ = torch_npu.npu_rms_norm(x_, weight, eps)
    return x_, None, residual_
    """
    x_ = x + residual
    residual_ = x_
    x_, _ = torch_npu.npu_rms_norm(x_, weight, eps)
    return x_, None, residual_
```

python/sglang/srt/layers/sampler.py

确定性采样逻辑修改，将 Ascend 路径上的 torch.multinomial 替换为带种子的 multinomial_with_seed。

```
def _sample_from_logits(
    self,
    logits: torch.Tensor,
    sampling_info: SamplingBatchInfo,
    simple_sampling_case: bool,
    positions: torch.Tensor, # 新增 positions 参数用于种子计算
) -> torch.Tensor:
    """
    从温度缩放后的 logits 采样，支持 seed 控制的确定性采样。
    """
    if simple_sampling_case:
        probs = torch.softmax(logits, dim=-1)
        if sampling_info.sampling_seed is not None:
            # 使用种子进行确定性采样：将概率转为对数，再调用 multinomial_with_seed
            probabilities = probs.to(torch.float64).log_()
            batch_next_token_ids = multinomial_with_seed(
                probabilities, sampling_info.sampling_seed, positions
            ).view(-1)
        else:
            batch_next_token_ids = torch.multinomial(probs, num_samples=1).view(-1)
        return batch_next_token_ids.to(torch.int32)
    else:
        assert self.use_ascend_backend, "Only ascend backend supports sampling from logits"
        batch_next_token_ids = top_k_top_p_min_p_sampling_from_logits_ascend(
            logits,
            sampling_info.top_ks,
            sampling_info.top_ps,
            sampling_info.min_ps,
            sampling_info.need_min_p_sampling,
            sampling_info.sampling_seed,
            positions,
        )
        return batch_next_token_ids.to(torch.int32)
```

python/sglang/srt/server_args.py

将 ascend 加入确定性注意力后端列表和 radix cache 支持列表，并在确定性推理时保留 ascend 采样后端。

```
# 确定性推理可用的注意力后端列表，新增 ascend
DETERMINISTIC_ATTENTION_BACKEND_CHOICES = [
    "flashinfer", "fa3", "triton", "ascend" # ascend 加入支持
]
# 支持 radix cache 的确定性注意力后端列表，新增 ascend
RADIX_SUPPORTED_DETERMINISTIC_ATTENTION_BACKEND = [
    "fa3", "triton", "ascend"
```

```
]

# 在 _handle_deterministic_inference 方法中
if self.enable_deterministic_inference:
    # 对于采样后端，如果当前不是 ascend，则强制改为 pytorch
    if self.sampling_backend != "ascend":
        self.sampling_backend = "pytorch"
        logger.warning("Sampling backend is set to pytorch for deterministic inference.")
```

评论区精华

- 操作符替换范围讨论：Hexq0210 质疑 npu_mean_batch_invariant 等是否真正 batch-invariant，尤其是带 dim 的 sum。chx96642264 解释只替换带 dim 的 sum，因为无 dim 的 sum 用于调度长度计算（int 无不确定性），GPU 也类似不替换所有算子。已接受当前替换范围。
- NPU 操作符代码分离：Todobe 要求将所有 NPU 相关函数移出 batch_invariant_ops.py，单独放到 hardware_backend/npu/ 目录下以避免污染公共代码。已创建 npu_batch_invariant_ops.py 并完成迁移。
- 采样后端设置逻辑：Todobe 指出当 attention_backend 为 ascend 时不应强制将采样后端设为 pytorch，否则会覆盖 ascend 后端。最终采用 if self.sampling_backend != "ascend" 的条件保护，已合并。
  - 操作符 batch-invariant 性讨论 (correctness): 接受当前替换范围，维持只替换带 dim 的算子。
  - NPU 操作符代码分离 (design): 已创建 npu_batch_invariant_ops.py 并完成迁移。
  - 采样后端设置逻辑 (design): 已修改为保留 ascend 采样后端。

风险与影响

- 风险：
  1. 外部依赖风险：依赖 batch_invariant_ops 第三方库，该库可能未正式发布或存在兼容性问题，若操作符语义不一致将导致数值偏差。
  2. 性能影响：确定性模式下禁用了 allreduce 融合（enable_aiter_allreduce_fusion 等），可能降低通信效率；npu_add_rms_norm 拆分为两步增加了显存和计算开销。
  3. 覆盖完备性：当前只替换了核心操作符，如果其他非确定性操作符被调用但未覆盖，仍可能导致结果不一致。
  4. 缺少自动化测试：PR 未包含单元测试，确定性验证依赖手动运行截图中展示的用例，回归风险较高。 - 影响：用户影响：使用 --enable-deterministic-inference 的 NPU 用户现在可以获得确定性的输出，对调试和测试非常有价值；但需关闭图模式，限制了部分场景的加速。系统影响：启用时禁用若干融合优化，降低吞吐；同时启用 radix cache 可能增加显存压力。团队影响：维护了新的 hardware_backend/npu/batch_invariant_ops/ 子模块，需跟踪上游 batch_invariant_ops 变更。
- 风险标记：外部依赖 batch_invariant_ops，缺少测试覆盖 ，确定性模式性能影响

关联脉络

- PR #25768 [NPU]Replace ascend vision attn operator: 本 PR 在确定性推理中替换了 npu_fused_infer_attention_score，与先前替换视觉注意力操作符的模式一致，都是针对 NPU 后端的算子替换。
- PR #28436 [NPU] Use use_dsa to dispatch Ascend DSA attention: 该 PR 修复了 Ascend 注意力调度条件，与本 PR 的确定性推理注意力后端配置相关。