

PR #39404 完整报告

sgl-project/sglang

[NPU] Avoid device synchronization in Ascend sampling

合并时间: 2026-09-16 15:15

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39404>

执行摘要

- 一句话: Ascend 采样改主机侧判定, 消除每 token 设备同步
- 推荐动作: 建议精读, 但只需要 10 分钟左右: 改动量极小, 却是理解 Ascend 采样热路径与「设备标量判定 → 隐式同步」这一常见性能陷阱的好样本。重点关注三处设计决策: 一是 `npu_top_k_top_p_eligible` 的计算位置选在 `from_schedule_batch` (而非每次采样), 依赖「`top_k` 在采样期间不变」这一前提; 二是 `merge_batch` 用 AND、默认值取 False 的保守取向, 使漏传与合并只带来性能退化而不带来正确性风险; 三是新增测试用 `torch.all` 抛 `AssertionError` 做护栏的写法, 可作为同类优化的测试模板。需要在真实 NPU 上验证收益的团队, 建议补充带同步计数的 profiler 证据, 而不是依赖吞吐数字。

功能与动机

PR body 明确指出根因: 调用 `torch_npu.npu_top_k_top_p` 前需要校验 `top_k` 是否落在 `[1, 1024]`, 而 `top_ks` 此时已存放在 NPU 上, 「Although `torch.all()` produces a device scalar, using that scalar in a Python condition requires converting it to a host boolean」——这就在 per-token 采样路径引入隐式 NPU 到 CPU 同步, 且因该条件每个 decode step 都求值而反复打断异步执行、抬高采样延迟。与此同时 `top_k` 完全来自请求参数、采样期间不变, 因此「It can therefore be evaluated once on the CPU when constructing the batch metadata」, 这构成本次改造的全部动机: 只前移可用性判定, 不改采样数学。

实现拆解

1. 元数据层新增主机侧标志 (`python/sglang/srt/sampling/sampling_batch_info.py`):
dataclass 新增 `npu_top_k_top_p_eligible: bool = False`, 注释写明「Keeping this off the device avoids a scalar synchronization in the per-token sampling path」; 在 `from_schedule_batch` 中与 `is_all_greedy`、`need_top_p_sampling` 等既有布尔标志并列计算 `all(1 <= r.sampling_params.top_k <= 1024 for r in reqs)`, 全程只读 CPU 侧请求参数, 不触碰设备张量。`TOP_K_ALL (-1)` 天然不满足下界, 继续走通用路径, 因此行为等价性由区间表达式保证。
2. 批次生命周期语义 (同文件 `merge_batch`): 新增 `self.npu_top_k_top_p_eligible &= other.npu_top_k_top_p_eligible`, 与 `is_all_greedy` 的 AND 语义一致、与 `need_*_sampling` 的 OR 语义相反; `filter_batch` 不做任何「重新提升」, 即过滤掉非法请求后仍保持保守。PR body 明确这是有意为之: 「This cannot incorrectly dispatch unsupported values to the fused operator」。

3. 采样分派点改造 (python/sglang/srt/layers/sampler.py) :

Sampler._sample_from_logits 在调用 top_k_top_p_min_p_sampling_from_logits_ascend 时新增关键字参数 npu_top_k_top_p_eligible=sampling_info.npu_top_k_top_p_eligible; 被调函数签名新增同名参数 (默认 False) , 融合分支条件由 hasattr(torch_npu, "npu_top_k_top_p") and torch.all(...) 变为 hasattr(...) and npu_top_k_top_p_eligible, 运行时算子可用性检查保持不变, 两个条件仍是 AND。参数默认值取 False 属于「安全方向」: 漏传只会导致性能退化, 不会把非法 top_k 送进融合算子。

4. 测试配套: 新增 test/registered/unit/sampling/test_ascend_sampler.py (register_cpu_ci(est_time=1, suite="base-a-test-cpu")) , 类 TestAscendSamplerDispatch 通过 patch.object(sampler_module.torch, "all", side_effect=AssertionError(...)) 建立回归护栏, 并断言 eligible 为 True/False 时 npu_top_k_top_p 的调用与否; test/registered/unit/sampling/test_sampling_batch_info.py 扩展 test_merge_combines_flags 增加 npu_top_k_top_p_eligible 的 AND 断言, 新增 test_npu_top_k_top_p_eligibility_uses_request_params 覆盖 (1, 1024)、(1, 1025)、(4, TOP_K_ALL) 三组边界。

5. 无配置与部署改动: 没有新增用户参数、环境变量或 CLI 开关, 优化对 Ascend 采样后端自动生效; 非 Ascend 路径 (CUDA/ROCm/MUSA 等) 因默认值为 False 成为 no-op, 行为与改动前完全一致。

关键文件:

- python/sglang/srt/sampling/sampling_batch_info.py (模块 采样批次信息; 类别 source ; 类型 core-logic; 符号 SamplingBatchInfo, from_schedule_batch, merge_batch) : 新增主机侧判定字段并在 batch 构造阶段一次性计算, 同时定义合并期的 AND 保守语义, 是本次优化的语义核心。
- python/sglang/srt/layers/sampler.py (模块 采样器; 类别 source; 类型 core-logic; 符号 _sample_from_logits, top_k_top_p_min_p_sampling_from_logits_ascend) : 采样热路径的实际改动点: 把设备侧 torch.all 判定替换为主机标志透传, 并新增带保守默认值的函数参数。
- test/registered/unit/sampling/test_ascend_sampler.py (模块 采样器测试; 类别 test; 类型 test-coverage; 符号 TestAscendSamplerDispatch, test_top_k_dispatch_does_not_read_device_values_on_cpu) : 新增的分派路径单测, 用让 torch.all 抛 AssertionError 的方式建立「不得再读设备标量」的回归护栏, 是本 PR 最具借鉴价值的测试写法。
- test/registered/unit/sampling/test_sampling_batch_info.py (模块 批次信息测试; 类别 test; 类型 test-coverage; 符号 test_merge_combines_flags, test_npu_top_k_top_p_eligibility_uses_request_params) : 覆盖新增标志的取值边界与合并语义, 保证 (1,1024)、(1,1025)、TOP_K_ALL 三组边界以及 merge 的 AND 行为有回归保护。

关键符号: Sampler._sample_from_logits, top_k_top_p_min_p_sampling_from_logits_ascend, SamplingBatchInfo.from_schedule_batch, SamplingBatchInfo.merge_batch, TestAscendSamplerDispatch.test_top_k_dispatch_does_not_read_device_values_on_cpu, test_npu_top_k_top_p_eligibility_uses_request_params

关键源码片段

python/sglang/srt/sampling/sampling_batch_info.py

新增主机侧判定字段并在 batch 构造阶段一次性计算，同时定义合并期的 AND 保守语义，是本次优化的语义核心。

```
# python/sglang/srt/sampling/sampling_batch_info.py

@dataclasses.dataclass
class SamplingBatchInfo:
    # ... 其余字段略 ...
    device: str = "cuda"

    # Handle logit bias
    logit_bias: Optional[torch.Tensor] = None

    # 主机侧 (host) 判定: 本 batch 是否可以使用 torch_npu.npu_top_k_top_p。
    # 判定留在 CPU 上, 是为了避免在每 token 的采样路径里做隐式标量同步。
    npu_top_k_top_p_eligible: bool = False

    @classmethod
    def from_schedule_batch(cls, batch: ScheduleBatch, vocab_size: int):
        reqs = batch.reqs
        device = batch.device
        _pin = is_pin_memory_available(device)
        # ... temperatures / top_ps / top_ks / min_ps 等张量构造略 ...

        ret = cls(
            temperatures=temperatures,
            top_ps=top_ps,
            top_ks=top_ks,
            min_ps=min_ps,
            sampling_seed=sampling_seed,
            is_all_greedy=all(r.sampling_params.top_k <= 1 for r in reqs),
            is_any_greedy=any(r.sampling_params.top_k <= 1 for r in reqs),
            need_top_p_sampling=any(r.sampling_params.top_p != 1.0 for r in reqs),
            need_top_k_sampling=any(r.sampling_params.top_k != TOP_K_ALL for r in reqs),
            need_min_p_sampling=any(r.sampling_params.min_p > 0 for r in reqs),
            # 只读请求参数: 采样过程中 top_k 不会变化, 因此在这里算一次就够。
            # TOP_K_ALL (-1) 不满足下界, 会继续走通用采样路径。
            npu_top_k_top_p_eligible=all(
                1 <= r.sampling_params.top_k <= 1024 for r in reqs
            ),
            vocab_size=vocab_size,
            penalizer_orchestrator=penalizer_orchestrator,
            has_custom_logit_processor=has_custom_logit_processor,
            custom_params=custom_params,
            custom_logit_processor=merged_custom_logit_processor,
            device=device,
```

```

        logit_bias=logit_bias,
        return_sampling_masks=return_sampling_masks,
        sampling_mask_batch_indices=sampling_mask_batch_indices,
    )
    ret.adjusted_from_schedule_batch(batch, vocab_size)
    return ret

def merge_batch(self, other: "SamplingBatchInfo"):
    # ... 张量拼接与 sampling_mask 合并略 ...

    self.is_all_greedy &= other.is_all_greedy
    self.is_any_greedy |= other.is_any_greedy
    self.need_top_p_sampling |= other.need_top_p_sampling
    self.need_top_k_sampling |= other.need_top_k_sampling
    self.need_min_p_sampling |= other.need_min_p_sampling
    # AND 语义：合并后只要有一个请求的 top_k 落在 [1, 1024] 之外，
    # 整个 batch 就退回通用采样路径（与 need_*_sampling 的 OR 语义相反）。
    self.npu_top_k_top_p_eligible &= other.npu_top_k_top_p_eligible

    self.adjusted_merge_batch(other)

```

python/sglang/srt/layers/sampler.py

采样热路径的实际改动点：把设备侧 torch.all 判定替换为主机标志透传，并新增带保守默认值的函数参数。

```

# python/sglang/srt/layers/sampler.py

# Sampler._sample_from_logits 的 Ascend 分支：把 batch 构造阶段算好的标志透传下去，
# 采样器本身不再接触 top_ks 的设备侧取值范围判定。
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,
    npu_top_k_top_p_eligible=sampling_info.npu_top_k_top_p_eligible,
)
return batch_next_token_ids.to(torch.int32)

def top_k_top_p_min_p_sampling_from_logits_ascend(
    logits: torch.Tensor,
    top_ks: torch.Tensor,
    top_ps: torch.Tensor,
    min_ps: torch.Tensor,
    need_min_p_sampling: bool,
    sampling_seed: Optional[torch.Tensor],

```

```

positions: torch.Tensor,
# 默认 False: 任何未显式传入标志的调用点都会退回通用采样路径,
# 保证「漏传」只带来性能退化, 不会把非法 top_k 送进融合算子。
npu_top_k_top_p_eligible: bool = False,
):
    """A top-k, top-p and min-p sampling implementation for ascend npu with torch_npu interface.
    """

    # torch_npu.npu_top_k_top_p 要求 top_k 取值范围为 [1, 1024]。
    # 旧实现是 torch.all((top_ks <= 1024) & (top_ks >= 1)): top_ks 已在 NPU 上,
    # torch.all 产生设备标量, 在 Python 条件里求值会触发隐式 NPU 到 CPU 同步,
    # 而该分支每个 decode step 都要走一次, 等于反复打断异步执行。
    # 现在条件只读主机侧布尔值, 同步彻底消失; 算子可用性检查保持不变。
    if hasattr(torch_npu, "npu_top_k_top_p") and npu_top_k_top_p_eligible:
        logits_top_k_top_p = torch_npu.npu_top_k_top_p(logits, top_ps, top_ks)
        probs_top_k_top_p = logits_top_k_top_p.softmax(dim=-1)

        if need_min_p_sampling:
            min_p_thresholds = probs_top_k_top_p.max(dim=-1) * min_ps
            min_p_mask = probs_top_k_top_p < min_p_thresholds.view(-1, 1)
            probs_top_k_top_p.masked_fill_(min_p_mask, 0.0)

        if sampling_seed is None:
            batch_next_token_ids = torch.multinomial(probs_top_k_top_p, num_samples=1)
        else:
            logprobs_top_k_top_p = probs_top_k_top_p.to(torch.float64)
            del probs_top_k_top_p
            logprobs_top_k_top_p.log_()
            batch_next_token_ids = multinomial_with_seed(
                logprobs_top_k_top_p, sampling_seed, positions
            )
    else:
        # 通用路径保持不变。注意这里会就地改写 top_ks (TOP_K_ALL 改写为 vocab size) ,
        # 主机侧标志在 batch 构造时已定型, 因此不会被这种就地改写反向影响。
        probs = torch.softmax(logits, dim=-1)
        probs_sort, probs_idx = probs.sort(dim=-1, descending=True)
        # ... top-k / top-p / min-p 掩码与多项式采样略 ...

```

test/registered/unit/sampling/test_ascend_sampler.py

新增的分派路径单测, 用让 torch.all 抛 AssertionError 的方式建立「不得再读设备标量」的回归护栏, 是本 PR 最具借鉴价值的测试写法。

```
# test/registered/unit/sampling/test_ascend_sampler.py
```

```
register_cpu_ci(est_time=1, suite="base-a-test-cpu")
```

```

class TestAscendSamplerDispatch(unittest.TestCase):
    def test_top_k_dispatch_does_not_read_device_values_on_cpu(self):
        logits = torch.tensor([[0.1, 0.2, 0.3, 0.4]])

```

```

top_ks = torch.tensor([2], dtype=torch.int32)
top_ps = torch.ones(1)
min_ps = torch.zeros(1)
positions = torch.zeros(1, dtype=torch.int64)

# 分别验证 eligible=True / False 两个分支，避免只覆盖融合路径
for eligible in (True, False):
    with self.subTest(eligible=eligible):
        npu_top_k_top_p = MagicMock(return_value=logits)
        with (
            patch.object(
                sampler_module,
                "torch_npu",
                SimpleNamespace(npu_top_k_top_p=npu_top_k_top_p),
                create=True,
            ),
            # 关键回归护栏：如果 Ascend 分派路径再次触碰设备张量判定
            # （例如重新引入 torch.all(top_ks <= 1024)），这里会直接抛
            # AssertionError 让测试失败，从机制上阻止同步被重新引入。
            patch.object(
                sampler_module.torch,
                "all",
                side_effect=AssertionError("device predicate was inspected"),
            ),
        ):
            result = (
                sampler_module.top_k_top_p_min_p_sampling_from_logits_ascend(
                    logits.clone(),
                    top_ks.clone(),
                    top_ps,
                    min_ps,
                    False,
                    None,
                    positions,
                    npu_top_k_top_p_eligible=eligible,
                )
            )

            self.assertEqual(tuple(result.shape), (1,))
            # 融合算子是否被调用，完全由主机标志决定
            self.assertEqual(npu_top_k_top_p.called, eligible)

```

评论区精华

本次 PR 的 review comments 为 0，评审意见只有 ping1jing2 的两次空 body APPROVED，实质讨论集中在 Issue 评论区：

作者 xiedeyantu: "Two requests have already been verified on the 910B4 machine, and the gen throughput of both requests is basically the same." (附

DeepSeek-V4-Flash 上 `top_k=20` 与默认参数两条 curl 请求)

作者 xiedeyantu: "This CI [PR Test (NPU) / base-b-test-4-npu-a3 / run (0)] failure is unrelated to the change. It occurs during Python module import while initializing the test environment, before any Ascend sampling code or test logic is executed."

可见两点：其一，作者给出的收益证据是「吞吐基本一致」+ 消除同步的理论论证，属于不引入回退的验证而非量化的性能提升证明；其二，CI 多次 `rerun-failed-ci` 后的失败被归因于测试环境 Python 模块导入阶段，与采样代码无逻辑关联。批次合并的 AND 语义、filter 不提升的保守策略均只在 PR body 中自述，没有 reviewer 交叉质询。

- 优化收益的实测验证方式 (performance): 作者以「吞吐基本一致」作为不回退证据，评审未进一步要求同步计数或延迟量化数据；真实收益仍缺少系统性 benchmark 支撑。
- CI 失败归因与重跑 (testing): 归因为测试环境导入问题，与本变更无逻辑关联；NPU 相关用例未提供真实设备上的采样分派验证结论。
- 批次生命周期中的保守资格判定 (design): 作者明确选择保守方向，牺牲部分性能换取「绝不把不支持的 `top_k` 送入融合算子」的正确性保证；该设计被评审接受但未经交叉验证。

风险与影响

- 风险：

1. 就地改写与固定标志的交互：通用分支中 `top_ks.masked_fill_(topk_all_mask, probs.shape[1])` 会把 TOP_K_ALL 就地改成 vocab size，而主机标志在 batch 构造时已定型、之后不再重算。方向是安全的（不会被「改写成合法区间」后误判为可用），但也没有任何路径能把标志从 False 提升为 True。
2. filter_batch 不重算导致长期保守：若批次中仅个别请求的 `top_k` 越界，只要它们还在 batch 内融合路径就不可用；被过滤后依然不恢复，长生命周期 batch 可能长期停留在通用采样路径。影响限于 Ascend 侧性能，不影响输出正确性。
3. 收益缺少端到端验证：新增单测注册在 CPU 套件 (base-a-test-cpu)，只能验证分派分支走向，无法度量真实同步消除；作者仅在 910B4 上用两条请求做手工验证，样本量小，且「吞吐基本一致」意味着收益更偏延迟与异步流水特性，而非吞吐提升，后续若要宣称收益需要更系统的 benchmark。
4. 回归护栏覆盖面有限：`patch.object(sampler_module.torch, "all", side_effect=AssertionError(...))` 只能拦住重新引入 `torch.all` 的写法；若未来改用 `.item()`、`torch.min`、`torch.any` 等方式读回设备标量，测试不会失败。
5. 共享结构体的影响面：`SamplingBatchInfo` 被所有后端共用，新字段会参与 `dataclasses.replace/copy_for_forward` 与 `merge` 语义；默认 False 使非 Ascend 路径成为 no-op，但后续任何新增构造或复制路径都需记得维护该标志，否则 Ascend 侧会静默退化。
 - 影响：影响范围集中在 Ascend/NPU 部署（如 910B 系列），且无需任何配置即可生效。对用户而言，采样语义（`top-k`、`top-p`、`min-p`、概率归一化、多项式采样、`sampling_seed` 行为）完全不变，不存在采样结果回归或参数兼容性问题。对系统而言，收益是移除了每 decode step 一次的设备与主机同步点，有利于 NPU 异步执行流水与采样延迟的稳定性；但由于实测吞吐持平，短期不宜期待明显吞吐提升。对团队而言，本 PR 沉淀了一个可复用范式：把「算子可用性 / 取值范围」这类静态判定前移到 batch 构

造阶段、危险默认值取保守方向、合并用 AND 语义，并配一条以 mock 抛异常实现的防回归护栏；该模式可推广到其他非 CUDA 后端的分派门槛校验。评审强度偏低（两次空 body APPROVED、0 条 review comments）意味着这些语义主要靠作者自证，读者需要自行确认。 - 风险标记：热路径同步语义变更，保守标志可能长期停留通用路径，缺少真实 NPU 端到端性能验证，回归护栏仅覆盖 torch.all 写法，共享结构体新增字段需维护

关联脉络

- PR #37134 [ROCm] Fix EAGLE spec-decode verify silently sampling greedy on HIP: 同属非 CUDA 加速器后端的采样分派治理：一个修复 HIP 上 verify 静默贪心（走错分支），一个消除 Ascend 采样判定引入的设备同步（分支判定方式本身的成本），两者都围绕采样后端分派路径的正确性与效率。
- PR #39561 [NPU] [bugfix] Fix undefined require_attn_tp_allgather: 同为 NPU/Ascend 后端路径的持续加固，反映该后端从「能跑通」向「热路径稳定可控」推进的同一演进线。
- PR #38402 fix(npu): fix hybrid KV transfer with PP prefill in PD disaggregation: 同为 Ascend 侧改动并附带注册单测（NPU/PD 场景），与本 PR 一样强调在 NPU 上保持行为等价与测试可回归。
- PR #39000 [Router] Fleet-wide sampling contract 1/3: the config surface: 概念上有交集：那条 PR 把采样参数约束下沉为路由侧配置面，本 PR 则是把同一个 top_k 范围约束在引擎侧前移到 CPU 计算，属于「采样契约」在不同层次上的实现。