

PR #36630 完整报告

sgl-project/sglang

[Sampling] Capture masks from sampler support

合并时间: 2026-09-03 08:13

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/36630>

执行摘要

- 一句话: sampler 原生捕获支持集, 修复 sampling mask 语义偏差
- 推荐动作: 值得精读。重点关注三处设计决策: 一是 capture 与融合采样共享 cutoff/tie/joint-support 语义的不变量边界, 这是本 PR 正确性的核心; 二是 select_capture_rows 的行级 opt-in 裁剪如何把 batch 级 any() 语义的捕获成本收敛到实际请求行数; 三是针对 FlashInfer 阈值型过滤与 PyTorch 位置型过滤差异的双重重建测试策略, 值得在后续后端适配中复用。

功能与动机

PR body 明确指出旧实现的缺陷: "Sampling-mask replay must return the exact final support $S_t = \{i \mid w_t[i] > 0\}$ and the sampled token logprob under that filtered distribution", 而旧实现是从采样后的原始概率重建 top-k/top-p/min-p 支持集, 再用补加采样 token 的方式修复重建结果。后端 cutoff、tie 与 filter-order 语义的差异会让重建结果与采样器真实使用的支持集不一致, 补 token 只能保证 action 在集合内, 无法恢复缺失 token 或剔除多余 token, 导致 replay 在错误的 action space 上做归一化。

实现拆解

1. 入口重构 (python/sglang/srt/layers/sampler.py 的 Sampler.forward): 将 _attach_sampling_mask_to_output 的调用从“采样后立即执行”移动到 token 同步 (_sync_token_ids_across_tp) 之后, greedy 分支的 mask 附加也统一到该时点; 当 SYNC_TOKEN_IDS_ACROSS_TP 或 grammar 约束可能替换生产者选中的 token 时, 用 _replace(selected_weight=None) 丢弃失效的 selected_weight, 避免基于被替换 token 计算 logprob。
2. 捕获数据结构: 新增 _SamplingMaskCapture(NamedTuple), 持有 weights (后过滤权重)、token_ids (PyTorch 路径的排序 token ID, FlashInfer 为 None)、selected_weight (采样 token 在过滤分布下的权重) 与 batch_rows (原始 batch 行映射)。支持集严格定义为 $w_t > 0$, $\log(w[y] / \sum(w))$ 从同一 capture 计算。
3. FlashInfer 路径: 融合采样 top_k_top_p_sampling_from_probs(..., filter_apply_order="joint") 保持不动, capture 用独立的 top_k_renorm_prob / top_p_renorm_prob 分步重归一化并取交集来复现 joint 语义——阈值处并列的 token 全部保留, 支持集可超过 top_k; min-p 则按 top-k $\rightarrow$ top-p $\rightarrow$ min-p 顺序捕获。代码注释中明确了正确性不变量: 融合采样器与重归一化原语必须共享 cutoff、tie、joint-support 语

义。

4. 行级 opt-in 与内存收敛：引入 `select_capture_rows` 闭包，`capture` 只物化 `return_sampling_masks` 中为 `True` 的行（FlashInfer 捕获核收到 `[R, V]` 而非 `[B, V]`）；全行 opt-in 时直接复用原张量，避免整批 `index_select` 拷贝。
`_attach_sampling_mask_to_output` 中 `support_mask` 改为 `weights.sum(dim=-1, dtype=torch.float32)` 计算，避免再生成 `masked` 浮点临时张量。
5. 配套清理与测试：`python/sglang/srt/sampling/sampling_batch_info.py` 删除不再使用的 `sampling_mask_max_top_k` 字段及其构造 /merge 记账；
`test/registered/sampling/test_sampling_mask.py` 新增 `TestSamplingMaskCapture` 聚焦单元测试（FlashInfer joint tie 边界、行级物化、PyTorch 压缩），并把端到端断言从“至多 `top_k + 1`”改为精确支持集比对（FlashInfer 按阈值 tie 重建、PyTorch 按位置截断重建）。

关键文件：

- `python/sglang/srt/layers/sampler.py`（模块 采样器；类别 source；类型 core-logic；符号 `_SamplingMaskCapture`, `select_capture_rows`, `_sample_from_probs`, `_attach_sampling_mask_to_output`）：核心实现文件。新增 `_SamplingMaskCapture`, `select_capture_rows`, 重构 `_sample_from_probs` 使其在请求时直接返回与采样器语义一致的捕获数据，并将 `mask` 附加时机移到 `token` 同步之后；同时删除旧的事后重建 `_compute_sampling_mask_from_probs` 与采样 `token` 修补逻辑。
- `test/registered/sampling/test_sampling_mask.py`（模块 采样测试；类别 test；类型 test-coverage；符号 `TestSamplingMaskCapture`, `test_flashinfer_joint_cutoff_ties_match_capture`, `test_flashinfer_capture_only_materializes_requested_rows`, `test_pytorch_capture_compacts_requested_rows`）：测试配套核心。新增 `TestSamplingMaskCapture` 单元测试覆盖 FlashInfer joint tie 边界、捕获行级物化与 PyTorch 行压缩，并把端到端断言从原来的 `top_k+1` 容忍改为精确支持集重建比对，直接钉住本 PR 的正确性不变量。
- `python/sglang/srt/sampling/sampling_batch_info.py`（模块 采样器；类别 source；类型 refactor；符号 `SamplingBatchInfo`）：配套清理。删除不再使用的 `sampling_mask_max_top_k` 字段及其在 `from_schedule_batch` 与 `merge_batch` 中的构造 / 合并记账，因为新捕获路径不再需要该上限状态。

关键符号：`_sample_from_probs`, `_attach_sampling_mask_to_output`, `_attach_greedy_sampling_mask_to_output`, `select_capture_rows`, `_compute_sampling_mask_from_probs`（已删除）

关键源码片段

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

核心实现文件。新增 `_SamplingMaskCapture`, `select_capture_rows`, 重构 `_sample_from_probs` 使其在请求时直接返回与采样器语义一致的捕获数据，并将 `mask` 附加时机移到 `token` 同步之后；同时删除旧的事后重建 `_compute_sampling_mask_from_probs` 与

采样 token 修补逻辑。

```
class _SamplingMaskCapture(NamedTuple):
```

```
    """保存过滤后的权重及与原始 batch 行的映射，供 mask 输出使用。"""
```

```
    weights: torch.Tensor # 后过滤权重；支持集严格定义为 weights > 0
```

```
    token_ids: Optional[torch.Tensor] # PyTorch 路径的排序 token ID, FlashInfer 为 None
```

```
    selected_weight: Optional[torch.Tensor] # 采样 token 在过滤分布下的权重
```

```
    batch_rows: torch.Tensor # 原始 batch 行索引，用于把捕获结果散回原位
```

```
    # _sample_from_probs 内：仅在请求 mask 时开启捕获，且只捕获 opt-in 的行。
```

```
    sampling_mask_capture = None
```

```
    capture_rows = None
```

```
    capture_all_rows = False
```

```
    if return_sampling_mask:
```

```
        # return_sampling_mask 是 batch 级 any() 语义；若不在此裁剪，单个请求
```

```
        # 开启 mask 就会让整个 batch 物化 [B, V] 权重，开销与 opt-in 行数脱钩。
```

```
        capture_rows_list = [
```

```
            i for i, should_return in enumerate(sampling_info.return_sampling_masks or [])
```

```
            if should_return
```

```
        ]
```

```
    if not capture_rows_list:
```

```
        raise RuntimeError(
```

```
            "Sampling-mask capture requested without any opted-in batch rows."
```

```
        )
```

```
    capture_rows = torch.tensor(capture_rows_list, device=probs.device, dtype=torch.long)
```

```
    capture_all_rows = capture_rows_list == list(range(probs.shape[0]))
```

```
def select_capture_rows(tensor: torch.Tensor) -> torch.Tensor:
```

```
    # 全行 opt-in 时直接复用原张量，省去一次整批 index_select 拷贝。
```

```
    assert capture_rows is not None
```

```
    return tensor if capture_all_rows else tensor.index_select(0, capture_rows)
```

```
    # FlashInfer joint 路径：融合采样保持原样，capture 用独立重归一化原语重建。
```

```
    batch_next_token_ids = top_k_top_p_sampling_from_probs(
```

```
        probs.contiguous(),
```

```
        sampling_info.top_ks,
```

```
        sampling_info.top_ps,
```

```
        filter_apply_order="joint",
```

```
    )
```

```
    if return_sampling_mask:
```

```
        # 正确性不变量：融合的 joint 采样器与下面的独立 top-k/top-p
```

```
        # 重归一化原语必须共享 cutoff、tie 与 joint-support 语义，
```

```
        # 否则捕获的正支持集无法精确描述采样器的 action space。
```

```
        capture_probs = select_capture_rows(probs)
```

```
        capture_top_ks = select_capture_rows(sampling_info.top_ks)
```

```
        capture_top_ps = select_capture_rows(sampling_info.top_ps)
```

```
        capture_tokens = select_capture_rows(batch_next_token_ids)
```

```
        filtered_probs = capture_probs
```

```
        if sampling_info.need_top_k_sampling:
```

```

        filtered_probs = top_k_renorm_prob(capture_probs, capture_top_ks)
    if sampling_info.need_top_p_sampling:
        top_p_probs = top_p_renorm_prob(capture_probs, capture_top_ps)
        if filtered_probs is capture_probs:
            filtered_probs = top_p_probs
        else:
            # 与 joint 语义一致：两个过滤器的保留集取交集。
            filtered_probs.masked_fill_(top_p_probs <= 0, 0)
    selected_weight = torch.gather(
        filtered_probs, 1, capture_tokens.long().view(-1, 1)
    ).squeeze(1)
    sampling_mask_capture = _SamplingMaskCapture(
        weights=filtered_probs,
        token_ids=None,
        selected_weight=selected_weight,
        batch_rows=capture_rows,
    )

# forward 内：mask 附加必须在 token 同步之后，因为同步可能替换生产者选中的 token。
if return_sampling_mask:
    if sampling_info.is_all_greedy:
        self._attach_greedy_sampling_mask_to_output(
            logits_output, sampling_info, batch_next_token_ids
        )
    else:
        assert sampling_mask_capture is not None
        if SYNC_TOKEN_IDS_ACROSS_TP or sampling_info.grammars:
            # token 同步或语法约束可能替换生产者选中的 token，
            # 捕获的 selected_weight 已失效，置 None 让上层重新计算。
            sampling_mask_capture = sampling_mask_capture._replace(
                selected_weight=None
            )
        self._attach_sampling_mask_to_output(
            logits_output,
            sampling_info,
            batch_next_token_ids,
            sampling_mask_capture,
        )

```

test/registered/sampling/test_sampling_mask.py

测试配套核心。新增 TestSamplingMaskCapture 单元测试覆盖 FlashInfer joint tie 边界、捕获行级物化与 PyTorch 行压缩，并把端到端断言从原来的 top_k+1 容忍改为精确支持集重建比对，直接钉住本 PR 的正确性不变量。

```

@unittest.skipIf(is_hip(), "FlashInfer is not available on ROCm")
def test_flashinfer_joint_cutoff_ties_match_capture(self):
    batch_size = 256
    top_k = 2
    top_p = 0.45

```

```

# 两个过滤器都在 0.2 处截断，tie 的 token 必须全部保留，
# 即使这会让支持集条目数超过 top_k。
base_probs = torch.tensor([[0.4, 0.2, 0.2, 0.1, 0.1]], device="cuda")
probs = base_probs.repeat(batch_size, 1)

# 独立推导阈值型 joint 支持集，与捕获结果比对。
sorted_probs = base_probs[0].sort(descending=True).values
top_k_cutoff = sorted_probs[top_k - 1]
mass_before = sorted_probs.cumsum(dim=-1) - sorted_probs
top_p_cutoff = sorted_probs[mass_before <= top_p][-1]
expected_support = (base_probs[0] >= top_k_cutoff) & (
    base_probs[0] >= top_p_cutoff
)
expected_ids = expected_support.nonzero(as_tuple=True)[0].tolist()
self.assertEqual(expected_ids, [0, 1, 2])

sampling_info = SimpleNamespace(
    sampling_seed=None,
    need_top_k_sampling=True,
    need_top_p_sampling=True,
    need_min_p_sampling=False,
    top_ks=torch.full((batch_size,), top_k, dtype=torch.int32, device="cuda"),
    top_ps=torch.full((batch_size,), top_p, device="cuda"),
    min_ps=torch.zeros(batch_size, device="cuda"),
    return_sampling_masks=[True] * batch_size,
)
with patch(
    "sglang.srt.layers.sampler.get_exec",
    return_value=SimpleNamespace(
        kernel=SimpleNamespace(sampling_backend="flashinfer")
    ),
):
    sampled, capture = self.sampler._sample_from_probs(
        probs,
        sampling_info,
        positions=torch.zeros(batch_size, dtype=torch.int64, device="cuda"),
        simple_sampling_case=False,
        return_sampling_mask=True,
    )

self.assertIsNotNone(capture)
self.assertEqual(capture.batch_rows.cpu().tolist(), list(range(batch_size)))
actual_support = capture.weights > 0
self.assertTrue(
    torch.equal(actual_support, expected_support.expand_as(actual_support))
)
# 关键断言：tie 使支持集条目数超过 top_k，且采样 token 必然落在支持集内。
self.assertGreater(int(actual_support[0].sum().item()), top_k)
self.assertTrue(

```

```
bool(actual_support.gather(1, sampled.view(-1, 1)).all().item())
)
```

评论区精华

Reviewer nanjiangwill 提出 4 条关键意见，作者 ByronHsu 逐一回应并落实：

- 捕获开销应按行裁剪：return_sampling_mask 是 batch 级 any() 语义，一个 opt-in 请求会导致 FlashInfer 为整个 batch 物化 [B, V] 权重。ByronHsu 在 1849430406 中引入 batch_rows 映射与 select_capture_rows，FlashInfer 捕获核现在只收到 [R, V]，全行 opt-in 时也避免了冗余 index_select。
- 不变量需要文档化并用测试钉住：nanjiangwill 指出“fused joint sampling 与独立 top-k/top-p 重归一化原语共享 cutoff、tie、joint-filter 语义”是本路径的正确性边界。ByronHsu 在 51b3c59f9e 中将该不变量写进代码注释，并添加合成 tie 测试。
- 保留合成 tie 边界测试：nanjiangwill 认为真实模型输出几乎不会产生精确 cutoff tie，仅靠 live-model 测试无法确定性覆盖。ByronHsu 用概率 [0.4, 0.2, 0.2, 0.1, 0.1]、top_k=2、top_p=0.45 验证 tie 产生支持集 {0, 1, 2}。
- support_mass 计算优化：nanjiangwill 建议用 weights.sum(dim=-1, dtype=torch.float32) 替代额外 masked 浮点临时张量。ByronHsu 在 f22763062a 应用，并保留 weights > 0 用于稀疏支持集打包。
- 删除未用状态：nanjiangwill 指出固定容量传输应自行校验实际长度，sampling_mask_max_top_k 不再需要。ByronHsu 同意并删除该字段及 merge 记账。
 - FlashInfer 捕获不变量需文档化并用合成 tie 测试钉住 (design): ByronHsu 在 51b3c59f9e 将该不变量写入代码注释，并添加合成 tie 回归测试验证 [0.4, 0.2, 0.2, 0.1, 0.1] + top_k=2 + top_p=0.45 下支持集为 {0, 1, 2}。
 - capture 成本应按 opt-in 行裁剪而非整 batch 物化 (performance): ByronHsu 在 1849430406 引入 batch_rows 映射与 select_capture_rows，FlashInfer 捕获核只处理 [R, V]，_attach_sampling_mask_to_output 直接消费压缩 capture，全行路径也避免了冗余拷贝。
 - 保留合成 tie 边界测试，避免只依赖 live-model 输出 (testing): ByronHsu 添加 test_flashinfer_joint_cutoff_ties_match_capture，用合成概率确定性覆盖 tie 边界，并跳过 ROCm 上的 FlashInfer 用例。
 - support_mass 计算避免额外 dense 临时张量 (performance): ByronHsu 在 f22763062a 应用该写法，weights > 0 谓词仍用于稀疏支持集打包，CUDA 聚焦测试与 attachment/logprob 校验均通过。
 - 移除未使用的 sampling_mask_max_top_k 状态 (design): ByronHsu 确认本 PR 不引入固定容量传输，现有 disaggregation 元数据缓冲已显式校验长度，并删除 sampling_mask_max_top_k 字段及 merge 记账。

风险与影响

- 风险：

1. 后端语义漂移风险：FlashInfer 融合采样与 `top_k_renorm_prob / top_p_renorm_prob` 独立原语之间的一致性依赖底层 kernel 的 `cutoff` 与 `tie` 语义保持稳定；若 FlashInfer 更新过滤行为，`capture` 会再次与采样器不一致。本 PR 用合成测试钉住了当前语义，但未做 kernel 版本兼容性验证。
2. 行为变更风险：采样 token 落在正支持集之外现在视为 `internal invariant failure`（硬 assert 或错误），而不是旧实现的修补行为。若某些后端数值边界此前依赖“修补”兜底，升级后将直接暴露异常。
3. `selected_weight` 失效路径：`SYNC_TOKEN_IDS_ACROSS_TP` 或 `grammar` 存在时 `selected_weight` 被置 `None`，依赖上层基于同步后 token 重新计算 `logprob`；若该路径未正确实现，会出现 `logprob` 缺失。
4. PyTorch 路径性能：PyTorch 采样本身仍需全 batch 排序 / 过滤，行级压缩发生在采样之后，`opt-in` 行少时并不能节省采样计算，只是节省了 `capture` 物化成本。
5. 字段移除兼容性：`sampling_mask_max_top_k` 被删除，若有其他未发现的模块依赖该字段会断裂，PR 内未见遗留引用。
 - 影响：对用户：`return_sampling_mask` 的输出字段保持兼容，但 `mask` 内容更精确——`tie` 会被保留、不再有 `+1` 修补 token；默认推理路径完全不受影响。对 RL 训练回放管线：获得精确支持集与过滤分布下的 `logprob`，回放归一化分母正确。对系统：`capture` 计算只在 `opt-in` 时发生且按行裁剪为 `[R, V]`，单请求开销可控。对团队：确立了“sampler 是 action space 唯一事实来源”的设计原则，并为后续 `async D2H` 传输与 `overlap` 调度 PR 清理了状态。
 - 风险标记：采样核心路径变更，后端语义一致性依赖，越界采样由修补改为硬失败，token 同步路径 `logprob` 降级，移除公共字段 `sampling_mask_max_top_k`

关联脉络

- PR #39164 `Generalize auxiliary outputs`: 弱关联：该 PR 重构了采样观测与辅助输出契约（`sampling_observer.py`、`auxiliary_output.py`），与本 PR 同属采样输出管线演进方向；本 PR 的 `sampling mask` 捕获也可视为 `aux output` 契约族的一部分。