# PR #33423 完整报告

sgl-project/sglang

Deterministic gumbel sampling: clamp u=1 so masked tokens can't be sampled

合并时间：2026-08-09 21:43

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33423

## 执行摘要

- 一句话：修复确定性 gumbel 采样 u=1 端点，杜绝采样被屏蔽 token
- 推荐动作：值得精读。核心是一行 clamp 变更，但其数学论证（哈希网格间距、半开区间端点、bit-identical 保证）可以作为数值稳定性修复的范例；测试用例通过预计算 (seed, position, column) 精确命中 0xFFFFFFFF 端点，避免了概率性测试的不稳定。关注点：multinomial_with_seed 的共享语义、clamp 上限与 float64 精度的关系，以及后续是否把同样端点保护迁移到其他哈希采样实现（如 EAGLE 专用采样）。

## 功能与动机

multinomial_with_seed 从 32 位哈希生成每个 (seed, position, column) 的均匀数并计算 gumbel 噪声 -log(-log(u))，当 hash 为 0xFFFFFFFF 时 u == 1.0 产生 +inf：一是被屏蔽 token 会被采样（+inf gumbel 加 -inf logprob 得 NaN，argmax 返回该列）；二是该桶主导整行（朴素有限钳制得到约 +708，而其他桶最大约 +22.2）。PR body 指出这同时也是 #30822 中已记录的 seeded EAGLE 采样 half-open-uniform 端点隐患，现于共享 gumbel 处一并修复。

## 实现拆解

### 1. 问题定位

在 python/sglang/srt/layers/sampler.py 的 multinomial_with_seed 中，gumbel 噪声由 x.log_().clamp_(min=...).neg_() 后再取一次 log 得到。x 来自 murmur_hash32 结果除以 uint32 max，覆盖 [0,1] 闭区间，因此 x == 1.0（hash 0xFFFFFFFF）时内层 -log(x) 为 0，外层 -log(0) 为 +inf。原实现只钳制了下界，端点未防护。

### 2. 一行钳制修复

把内层 clamp 增加上界 max=-(2.0**-32)：u == 1.0 桶的 -log(u) 被限制在哈希网格间距 2^-32，对应 gumbel 值 32•ln2 ≈ 22.18，恰为相邻桶的自然最大值。其余所有桶的数值位级不变，被屏蔽列的 -inf logprob 加上有限 gumbel 后仍为 -inf，argmax 不再可能选中被屏蔽列，且不改变确定性种子的采样结果分布。

### 3. 回归测试配套

新增 test/registered/sampling/test_deterministic_gumbel_u1.py（51 行），通过预计算的 (SEED=6469398791980356130, POSITION=7371, U1_COLUMN=248146) 精确复现

hash 0xFFFFFFFF：

- test_never_samples_masked_token：248320 词表中 CUTOFF 之后全部置 -inf，断言采样结果 < CUTOFF（在 main 分支因 NaN 路径失败）；
- test_u1_bucket_does_not_dominate：均匀行中仅将 u==1 列设为 logit -40，断言不被采样（在 tiny-style 钳制下因 +708 异常值失败）。 测试通过 register_cuda_ci / register_amd_ci 注册到 CUDA 与 AMD CI。

### 4. 配套说明

无配置、schema 或部署改动；sampling_from_probs_torch 与 Sampler._sample_from_logprobs 自动共享该修复。CI 过程中因 main 更新要求 rebase，合并 main 后 rerun 通过（1-gpu-5090）。

关键文件：

- python/sglang/srt/layers/sampler.py（模块 采样器；类别 source；类型 core-logic；符号 multinomial_with_seed）：确定性采样共享入口 multinomial_with_seed 的核心修复文件，一行 clamp 变更消除 u == 1.0 桶的 +inf gumbel 异常，影响所有 seeded 采样路径。
- test/registered/sampling/test_deterministic_gumbel_u1.py（模块 采样测试；类别 test；类型 test-coverage；符号 _sample, TestDeterministicGumbelU1, test_never_samples_masked_token, test_u1_bucket_does_not_dominate）：新增回归测试文件，通过预计算哈希值精确命中 0xFFFFFFFF 端点，覆盖两个失败模式，并注册 CUDA/AMD CI。

关键符号：multinomial_with_seed, _sample, test_never_samples_masked_token, test_u1_bucket_does_not_dominate

## 关键源码片段

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

确定性采样共享入口 multinomial_with_seed 的核心修复文件，一行 clamp 变更消除 u == 1.0 桶的 +inf gumbel 异常，影响所有 seeded 采样路径。

```
# multinomial_with_seed 是确定性采样（seeded gumbel trick）的共享实现，
# 同时被 Sampler._sample_from_logprobs 与 sampling_from_probs_torch 使用。
# 以下为该函数的核心计算段。

n, m = logprobs.shape
seed = seed.to(torch.uint64)
col_indices = torch.arange(m, device=logprobs.device)
hashed = murmur_hash32(seed, positions, col_indices)

# 保持 float64 计算以避免数值不稳定（沿用原实现注解）。
x = hashed.to(torch.float64) / torch.iinfo(torch.uint32).max

# x 是 [0, 1] 上的均匀样本，gumbel 噪声为 -log(-log(x))。
# 关键修复：hash == 0xFFFFFFFF 时 x == 1.0，若不加处理，-log(x) 为 0，
# 外层 -log(0) 得 +inf，与 -inf logprob（被屏蔽列）相加得到 NaN，
```

```
# argmax 会误选被屏蔽的 token；同时该桶会以约 +708 的异常值主导整行。
# 这里将 -log(x) 上界钳制到哈希网格间距 2^-32，使该桶 gumbel 恰为
# 32 * ln2 ≈ 22.18，与相邻桶的自然最大值一致，不再主导行分布；
# 其余桶的取值保持不变，确定性行为位级不变。
x.log_().clamp_(min=torch.finfo(x.dtype).min, max=-(2.0**-32)).neg_() # -log(x)
x.log_().neg_() # -log(-log(x)) == gumbel noise，即最终 gumbel 值

# 将 gumbel 噪声叠加到 logprobs 后取 argmax，完成确定性采样。
x.add_(logprobs.to(torch.float64))
return torch.argmax(x, dim=1, keepdim=True)
```

## test/registered/sampling/test_deterministic_gumbel_u1.py

新增回归测试文件，通过预计算哈希值精确命中 0xFFFFFFFF 端点，覆盖两个失败模式，并注册 CUDA/AMD CI。

```
# 该用例构造精确命中 murmur_hash32 == 0xFFFFFFFF 的 (seed, position, column)，
# 从而稳定复现 u == 1.0 端点问题，而不是依赖 2^-32 的偶发概率。
VOCAB = 248320
CUTOFF = 248077
# murmur_hash32(seed, position, 248146) == 0xFFFFFFFF
SEED, POSITION = 6469398791980356130, 7371
U1_COLUMN = 248146


def _sample(logits: torch.Tensor) -> int:
    probs = torch.softmax(logits, dim=-1)
    return int(
        sampling_from_probs_torch(
            probs,
            sampling_seed=torch.tensor([SEED], device="cuda"),
            positions=torch.tensor([POSITION], device="cuda"),
        ).item()
    )


class TestDeterministicGumbelU1(CustomTestCase):
    def test_never_samples_masked_token(self):
        # 将词表尾部列置为 -inf 屏蔽，采样结果必须落在屏蔽区之外；
        # 修复前 u == 1.0 桶的 +inf gumbel 会把该列变成 NaN 并被 argmax 选中。
        torch.manual_seed(0)
        logits = torch.randn(1, VOCAB, device="cuda", dtype=torch.float32) * 4
        logits[:, CUTOFF:] = float("-inf")
        self.assertLess(_sample(logits), CUTOFF)

    def test_u1_bucket_does_not_dominate(self):
        # 单独把 u==1.0 桶所在列降到 logit -40（softmax 概率约 1.7e-23，
        # 在 fp32 中仍可表示），修复前该列 gumbel 异常值 +708 会主导整行；
        # 钳制后其 gumbel 至多 22.18，与相邻桶同级，不再遮蔽模型分布。
        logits = torch.zeros(1, VOCAB, device="cuda", dtype=torch.float32)
```

```
    logits[:, U1_COLUMN] = -40.0
    self.assertNotEqual(
        _sample(logits), U1_COLUMN, "u==1 gumbel outlier overrode a ~-52 logprob"
    )
```

## 评论区精华

两位 reviewer（sshleifer、ispobock）均直接 APPROVED，sshleifer 仅给出 'LGTM'，无技术性评论交锋。PR body 中作者对钳制值的选择给出了完整数学论证——'Clamp the inner -log(u) at 2^-32 — the hash grid spacing — so the u == 1.0 bucket maps to gumbel = 32•ln2 ≈ 22.18, exactly the natural maximum of its neighboring bucket'，并说明其他桶 bit-identical。CI 侧 ispobock 两次发起 /rerun-test，第一次因分支与 main 分歧被 bot 拒绝，合并 main 后 rerun 在 1-gpu-5090 上通过。整体属于低争议、低风险变更。

- CI rerun 因分支 diverged 被拒，rebase 后通过 (other): 合并 main 后测试通过，无遗留问题。

## 风险与影响

- 风险：
  1. 共享采样路径影响面：multinomial_with_seed 是确定性采样的统一入口（同时被 sampling_from_probs_torch 与 Sampler._sample_from_logprobs 调用），虽然只改一行，但会改变所有 seeded 采样路径中 u == 1.0 桶的行为；不过该桶 gumbel 由 +inf/异常值变为 22.18 后与其他桶一致，从数学上不改变非端点桶的分布。
  2. 数值边界：钳制上限 -(2.0**-32) 依赖 float64 下 x.log() 的精度；代码已保持 float64 运算，次高桶 -log(1-2^-32) ≈ 2^-32 + 2^-65 不会被误钳制，需要持续测试保障。
  3. 测试覆盖：新增测试为 CUDA GPU 测试（device='cuda'），AMD CI 注册依赖对应 runner 可用性；仅覆盖 0xFFFFFFFF 单点，未覆盖 0x00000000（u == 0 端点）的回归测试。
  4. 兼容性：对未使用 deterministic seeded sampling 的用户无影响；对使用该模式的用户，同一 seed/position 下除端点桶外采样结果保持不变。 - 影响：影响范围：修复所有启用确定性 / 种子化采样（seeded gumbel trick）场景的正确性，包括 EAGLE、REST 等依赖 seed 一致性的推测解码路径以及一致性 / 确定性推理测试。按 PR body 估算，248k 词表下约每 17k 采样 token 会出现一次端点命中，修复前表现为偶发输出被屏蔽 token 或分布偏差；修复后彻底消除该端点 hazard。对团队而言，新增了精确构造哈希端点的回归测试模式，可复用于未来 murmur_hash32 相关改动；测试已纳入 CUDA/AMD CI。影响程度：中低（单行逻辑 + 新增测试），但修复的是隐蔽的正确性问题。 - 风险标记：确定性采样共享路径变更，数值端点依赖 float64 精度，测试仅覆盖 CUDA/ 单端点，影响所有 seeded 采样场景

## 关联脉络

- PR #34159 Fix deterministic inference all-reduce for tp>1: 同属确定性推理一致性修复主线，一个修复 TP>1 下 NCCL 确定性，本 PR 修复 gumbel 端点采样正确性。

- PR #34168 Add deterministic logprob-consistency test for inkling-small nvfp4: 同为确定性 logprob/ 采样一致性验证，说明仓库正在系统加固确定性采样与输出。