PR #45996 完整报告

vllm-project/vllm

[MRV2] Make FP32 Gumbel sampling more accurate

合并时间：2026-06-19 03:42

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45996

执行摘要

- 一句话：使用 log1p(-u) 提高 FP32 Gumbel 采样精度
- 推荐动作：值得精读。该 PR 展示了通过数值变换解决浮点精度限制的精巧优化，适用于所有使用 Gumbel-max 采样的场景。推荐关注 MRV2 演进的同学阅读。

功能与动机

The current FP32 Gumbel sampling kernel is not accurate enough. This PR improves the accuracy with a simple trick: use log1p(-u) instead of log(u), enabling higher resolution in determining the argmax winner.

实现拆解

1. 算法核心修改：在 vllm/v1/worker/gpu/sample/gumbel.py 的 gumbel_block_argmax 函数的 FP32 分支中，将 gumbel_noise = -tl.log(-tl.log(u)) 替换为 gumbel_noise = -tl.log(-tldevice.log1p(-u))。
2. clamp 常量调整：将全局常量 _FP32_TINY（0x1p-126）替换为 _TL_RAND_MIN（4.6566127342e-10），因为 tl.rand 实际返回的最小值是 4.6566e-10 而非 2^-126，避免 clamp 无效。
3. 新增统计测试套件：新建 tests/v1/worker/test_gpu_gumbel_sample.py，包含 test_sampling_matches_target_distribution（Z-score 检验）、test_full_vocab_distribution_fidelity（完整词汇分布频率检验）和 test_greedy_temperature_zero_returns_argmax（温度 0 时返回 argmax）。测试采用重尾分布（一个 dominant token，其余 18 logits 以下）专门暴露 fp32 精度短板。

关键文件：

- vllm/v1/worker/gpu/sample/gumbel.py（模块 采样核；类别 source；类型 core-logic；符号 _TL_RAND_MIN, gumbel_block_argmax）：核心采样核修改，包括精度变换和常量更新
- tests/v1/worker/test_gpu_gumbel_sample.py（模块 采样测试；类别 test；类型 test-coverage；符号 _make_heavy_tailed_counts, _counts_to_logits, _sample, _z_score）：新增完整统计测试套件，验证重尾分布下的采样保真度

关键符号：gumbel_block_argmax, gumbel_sample, _sample, _make_heavy_tailed_counts, test_sampling_matches_target_distribution

关键源码片段

vllm/v1/worker/gpu/sample/gumbel.py

核心采样核修改，包括精度变换和常量更新

```
@triton.jit
def gumbel_block_argmax(
    logits_ptr, logits_stride, block, mask, req_state_idx, token_idx,
    seeds_ptr, pos_ptr, temp_ptr, processed_logits_ptr, processed_logits_stride,
    processed_logits_col_ptr, vocab_size, BLOCK_SIZE: tl.constexpr,
    APPLY_TEMPERATURE: tl.constexpr, USE_FP64: tl.constexpr,
    PER_TOKEN_COL: tl.constexpr,
):
    # ... 前面逻辑省略 ...
    # 核心噪声生成（HEAD 版本）
    if USE_FP64:
        u = tl_rand64(gumbel_seed, block, includes_zero=False)
        gumbel_noise = -tl.log(-tl.log(u))
    else:
        u = tl.rand(gumbel_seed, block)
        u = tl.maximum(u, _TL_RAND_MIN)
        # 将大噪声尾映射到 u -> 0 区域以获得更高 fp32 精度
        # 使用 log1p(-u) 替代 log(-log(u))，避免在 u -> 1 时精度不足
        gumbel_noise = -tl.log(-tldevice.log1p(-u))

    # 应用 Gumbel 噪声
    logits = tl.where(mask, logits + gumbel_noise, float("-inf"))

    value, idx = tl.max(logits, axis=0, return_indices=True)
    return value, idx
```

tests/v1/worker/test_gpu_gumbel_sample.py

新增完整统计测试套件，验证重尾分布下的采样保真度

```
NUM_SAMPLES = 500_000
HEAD_LOG_GAP = 18.0
Z_TOLERANCE = 10.0

def _make_heavy_tailed_counts(seed: int = 1234) -> torch.Tensor:
    gen = torch.Generator(device=DEVICE).manual_seed(seed)
    counts = torch.randint(
        1, 4, (VOCAB_SIZE,), generator=gen, dtype=torch.int64, device=DEVICE
    )
    counts[0] = round(math.exp(HEAD_LOG_GAP)) # dominant token
    return counts

def test_sampling_matches_target_distribution(use_fp64: bool):
    counts = _make_heavy_tailed_counts()
    total = counts.sum().item()
```

```
    logits = _counts_to_logits(counts)
    sampled = _sample(logits, NUM_SAMPLES, use_fp64=use_fp64)
    # 检查 dominant token (index 0) 的采样比例是否在 Z_TOLERANCE 范围内
    p0 = counts[0].item() / total
    observed0 = (sampled == 0).sum().item()
    z0 = _z_score(observed0, p0 * NUM_SAMPLES, NUM_SAMPLES)
    assert abs(z0) < Z_TOLERANCE, f"Z-score {z0} out of tolerance"
    # 检查 tail aggregate (index >= 1) 的采样比例
    tail_expected = (counts[1:].sum().item() / total) * NUM_SAMPLES
    tail_observed = (sampled >= 1).sum().item()
    z_tail = _z_score(tail_observed, tail_expected, NUM_SAMPLES)
    assert abs(z_tail) < Z_TOLERANCE, f"Tail Z-score {z_tail} out of tolerance"
```

评论区精华

无实质审查讨论，变更由 njhill 快速批准合并。仅合并前有 mergify 提示 pre-commit 失败，已修复。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。影响范围仅限于 MRV2 FP32 Gumbel 采样路径，不影响 fp64 路径或其他采样器。新引入的 tldevice.log1p 需要 Triton 版本支持，已通过条件导入防御。clamp 常量 _TL_RAND_MIN 的逻辑与之前一致，但需确认 tl.rand 的最小值文档化。测试覆盖了 50 万次采样，统计上显著，但仍需关注生产环境中潜在的罕见样本偏差。
- 影响：用户透明，采样质量提升（更接近理论分布），无 API 变化。性能可能轻微变化（log1p 比 log 稍慢，但噪声生成路径相同数量级）。仅影响 MRV2 启用时的采样；fp64 路径可选但性能更差。团队维护负担低，代码简洁。
- 风险标记：采样精度改进，引入 tldevice.log1p 依赖

关联脉络

- PR #44446 [Model Runner V2] Migration to support quantized model by default [5/N]: 本 PR 属于 MRV2 系列改进，延续 44446 开始的 Model Runner V2 迁移