

PR #54282 完整报告

vllm-project/vllm

[Bugfix][Model Runner V2][Spec Decode] Decouple the draft's gumbel noise stream from the target's

合并时间: 2026-08-30 00:58

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54282>

执行摘要

- 一句话: 草稿与目标 Gumbel 噪声流解耦, 修复概率投机采样偏差
- 推荐动作: 值得精读。核心价值不在代码量, 而在三个可迁移的设计: 用 Philox 偏移盐廉价地切分噪声流; 用分布化检验 (而非逐 token 比对) 验证采样统计性质; 把隐式 +1 约定上移为各调用方显式声明。测试里“宽词表测不出、窄词表高样本才能分辨偏差”的敏感性分析 (chi2 12 vs 1500) 是统计测试设计的好范例, 适合作为投机解码采样正确性回归测试的模板。

功能与动机

PR 正文明确指出: 目标验证阶段对被拒草案 token 做残差重采样所用的 Gumbel 噪声, 与当初产生该草案的噪声来自同一个 Philox 偏移 (同 seed、同 pos), 即字节级相同的噪声。条件于提案赢得 argmax 后, 其余 token 的 Gumbel 值都被截断在该最大值之下——草案排名最高的 token 截断得最紧——于是残差重采样恰好低估这些 token, 输出分布不再等于目标分布。该问题只影响 `draft_sample_method="probabilistic"`; 默认 greedy 下 `draft_logits` 为 None, 草案从不调用 `gumbel_sample`, 不存在共享噪声向量。PR 还顺带修复了第二个潜在问题: 采样位置此前取自被 `max_model_len` 截断的缓冲区, 接近上下文上限时相邻两步草案可能复用同一个噪声向量。

实现拆解

实现按 5 步拆解:

1. 加盐解耦噪声流 (核心修复): 在 `vllm/v1/worker/gpu/sample/gumbel.py` 引入 `_DRAFT_NOISE_SALT = 1 << 30` (`tl.constexpr`), 为 `gumbel_noised_argmax` 增加编译期常量 `IS_DRAFTING`; 草稿采样时先对 `pos` 加盐再进入 `tl.randint(seed, pos)`, 使草稿流与目标验证 / 残差流使用互不相交的 Philox 偏移。公共入口 `gumbel_sample` 新增 `is_drafting` 参数, 逐层透传到 `gumbel_block_argmax` 与 `_gumbel_sample_kernel`, 所有 Triton 内核签名同步扩展。选择 `1 << 30` 是因为位置是 `int64` 且远小于 `2**30`, 两个流在可预见的范围内不会碰撞。
2. 采样位置约定上移: `vllm/v1/worker/gpu/spec_decode/speculator.py` 的 `sample_draft` 删除硬编码的 `positions + 1` (原为 Woosuk 注释的隐式约定), 改收调用方自报的 `sample_src_positions` 参数, 并显式传 `is_drafting=True`。各 `speculator` 在自身语境里声明位置语义: autoregressive 的 `_prefill` 计算 `positions + 1` (“位置 P 的隐藏态起草 P+2

- ，采样按被采样 token 的前一个位置取键，净调整 +1”）；multi_module_mtp 同样 positions + 1；dflash 由 sample_pos - 2 改为 sample_pos - 1（不再需要抵消内部 +1）；dflash2 与 dspark 在内核 /_sample_logits 中直接传 P - 1 并加 IS_DRAFTING=True。
3. 未截断采样位置缓冲：AutoRegressiveSpeculator 新增 self.sample_src_positions 张量，_prepare_decode_inputs_kernel 与 _update_draft_inputs_kernel 在每步草案推进时对该缓冲 +1；prepare_decode_inputs、update_draft_inputs 两个公共函数签名同步扩展并更新所有调用点。该缓冲独立于 input_buffers.positions，不经过 max_model_len 截断，从根上消除相邻草案步骤的噪声复用。
 4. 目标侧显式区分：vllm/v1/worker/gpu/sample/sampler.py 的 sample() 调用 gumbel_sample 时传 is_drafting=False，确认目标采样不加盐；rejection_sampler_utils.py 有 1 行配套微调（材料未完整展开，属于参数 / 注释层面的小改动）。
 5. 测试配套：tests/v1/spec_decode/test_rejection_sampler_utils.py 新增 _gumbel_drafted_tokens 辅助函数与 test_gumbel_drafted_rejection_sample_is_unbiased 回归测试，用窄词表（16 桶）× 20 万次试验构造“草案反向排序”的强分歧场景放大残差路径，去掉 is_drafting=True 时 position 0 的 chi2 从约 12 飙到约 1500（阈值约 70），证明宽词表测试无法分辨该偏差；tests/v1/worker/test_gpu_gumbel_sample.py 新增 test_drafting_uses_a_separate_noise_stream，验证加盐流与目标流在“离开 head 的抽样”上互不一致、两流都复现目标尾部质量、且草稿流可复现。

关键文件：

- vllm/v1/worker/gpu/sample/gumbel.py（模块 采样内核；类别 source；类型 core-logic；符号 _DRAFT_NOISE_SALT, gumbel_noised_argmax, gumbel_block_argmax, _gumbel_sample_kernel）：核心修复点：_DRAFT_NOISE_SALT 盐偏移与 IS_DRAFTING 编译期常量贯穿整个 Gumbel-max 采样内核链，是“草稿流与目标流解耦”的唯一机制。
- vllm/v1/worker/gpu/spec_decode/speculator.py（模块 投机解码；类别 source；类型 core-logic；符号 sample_draft, DraftModelSpeculator）：sample_draft 删除内部隐式 +1，把采样位置约定上移给各 speculator，并显式传 is_drafting=True，是本次重构的枢纽。
- vllm/v1/worker/gpu/spec_decode/autoregressive/speculator.py（模块 草稿生成；类别 source；类型 core-logic；符号 AutoRegressiveSpeculator, prepare_decode_inputs, update_draft_inputs, _prepare_decode_inputs_kernel）：新增独立且不截断的 sample_src_positions 缓冲，修复 max_model_len 截断导致的相邻草案步骤噪声复用；同时修改两个 Triton kernel 与两个公共函数签名。
- tests/v1/spec_decode/test_rejection_sampler_utils.py（模块 拒绝采样；类别 test；类型 test-coverage；符号 _gumbel_drafted_tokens, test_gumbel_drafted_rejection_sample_is_unbiased）：新增回归测试 test_gumbel_drafted_rejection_sample_is_unbiased，用窄词表高样本量复现生产草稿的噪声耦合，是证明修复有效的关键证据。
- tests/v1/worker/test_gpu_gumbel_sample.py（模块 采样内核；类别 test；类型 test-coverage；符号 test_drafting_uses_a_separate_noise_stream）：新增

test_drafting_uses_a_separate_noise_stream, 直接验证盐偏移生效且不扭曲分布, 是对加盐机制的单元级背书。

- vllm/v1/worker/gpu/spec_decode/multi_module_mtp/speculator.py (模块 多模块草稿; 类别 source; 类型 core-logic; 符号 _generate_drafts) : MTP 草稿生成改为显式 sample_src_positions = positions + 1 并在循环内逐步推进, 是位置约定上移的关键落点之一。
- vllm/v1/worker/gpu/spec_decode/dflash2/speculator.py (模块 投机解码; 类别 source; 类型 core-logic; 符号 _selector_walk_kernel) : 选择器遍历 Triton 内核 _selector_walk_kernel 调用 gumbel_noised_argmax 时补充 IS_DRAFTING=True, 草稿侧加盐覆盖 DFlash2 路径。
- vllm/v1/worker/gpu/spec_decode/dspark/speculator.py (模块 投机解码; 类别 source; 类型 core-logic; 符号 _sample_logits) : DSpark 的 _sample_logits 在 gumbel_sample 调用中补 is_drafting=True, 确保 DSpark 草案也走加盐流。
- vllm/v1/worker/gpu/spec_decode/dflash/speculator.py (模块 投机解码; 类别 source; 类型 core-logic; 符号 _generate_draft) : 因 sample_draft 不再内部 +1, 采样位置从 Q-2 修正为 Q-1, 是位置约定变更的直接下游修正。
- vllm/v1/worker/gpu/sample/sampler.py (模块 目标采样; 类别 source; 类型 core-logic; 符号 sample) : 目标侧采样显式传 is_drafting=False, 确认目标噪声流不加盐, 与草稿流形成对称契约。
- vllm/v1/worker/gpu/spec_decode/rejection_sampler_utils.py (模块 拒绝采样; 类别 source; 类型 core-logic) : 有 1 行配套微调 (材料未完整展开), 属于参数或注释层面的小改动, 与验证路径的噪声语义对齐。

关键符号: gumbel_sample, gumbel_noised_argmax, gumbel_block_argmax, sample_draft, prepare_decode_inputs, update_draft_inputs, _gumbel_drafted_tokens, test_gumbel_drafted_rejection_sample_is_unbiased, test_drafting_uses_a_separate_noise_stream

关键源码片段

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

核心修复点: `_DRAFT_NOISE_SALT` 盐偏移与 `IS_DRAFTING` 编译期常量贯穿整个 Gumbel-max 采样内核链, 是“草稿流与目标流解耦”的唯一机制。

```
# vllm/v1/worker/gpu/sample/gumbel.py (整理自 PR#54282 变更后版本)
```

```
# 草稿流噪声盐: 让草稿采样与目标验证 / 残差重采样使用互不相交的 Philox 偏移。
# 验证是概率比检验  $p(x) > u \cdot q(x)$ , 不是 Gumbel 耦合, 因此被拒样本的残差
# 重采样绝不能与生成它的草案共享同一个噪声向量, 否则残差会系统性低估
# 草案排名靠前的 token。位置是 int64 且远小于  $2^{**30}$ , 两个流不会碰撞。
_DRAFT_NOISE_SALT = tl.constexpr(1 << 30) if HAS_TRITON else (1 << 30)
```

```
@triton.jit
```

```
def gumbel_noised_argmax(
```

```

logits,
keys,
mask,
seed,
pos,
temp,
IS_DRAFTING: tl.constexpr, # 编译期常量: 草稿流则对 pos 加盐
USE_FP64: tl.constexpr,
APPLY_TEMPERATURE: tl.constexpr = True,
):
    """对 logits 做 Gumbel-max 采样返回 argmax; temp == 0 时为普通 argmax."""
    if temp != 0.0 and APPLY_TEMPERATURE:
        logits = logits / temp
    if USE_FP64:
        logits = logits.to(tl.float64)
    if temp != 0.0:
        if IS_DRAFTING:
            pos = pos + _DRAFT_NOISE_SALT # 关键改动: 草稿噪声流与目标解耦
            gumbel_seed = tl.randint(seed, pos)
            if USE_FP64:
                u = tl_rand64(gumbel_seed, keys, includes_zero=False)
                gumbel_noise = -tl.log(-tl.log(u))
            else:
                u = tl_rand32(gumbel_seed, keys, includes_zero=False)
                # log1p 保证 u -> 0 的获胜尾部在 fp32 下仍可分辨
                gumbel_noise = -tl.log(-tldevice.log1p(-u))
            logits = tl.where(mask, logits + gumbel_noise, float("-inf"))
    return tl.max(logits, axis=0, return_indices=True)

def gumbel_sample(
    logits, expanded_idx_mapping, temperature, seed, pos,
    apply_temperature: bool,
    is_drafting: bool, # 新增: 草稿采样侧必须显式传 True
    logits_cache=None, # [max_num_reqs, num_cols, vocab_size]
    logits_cache_col=None, # 标量或 [num_tokens]
    use_fp64: bool = False,
):
    # ..... 前处理省略, 内核启动处新增透传:
    _gumbel_sample_kernel[grid](
        ..., BLOCK_SIZE=BLOCK_SIZE,
        IS_DRAFTING=is_drafting, # 把加盐开关带给 Triton 内核
        APPLY_TEMPERATURE=apply_temperature,
        USE_FP64=use_fp64,
        PER_TOKEN_COL=per_token_col,
    )

```

[vllm/v1/worker/gpu/spec_decode/autoregressive/speculator.py](https://github.com/vllm-project/vllm/blob/main/vllm/worker/gpu/spec_decode/autoregressive/speculator.py)

新增独立且不截断的 `sample_src_positions` 缓冲，修复 `max_model_len` 截断导致的相邻草案步骤噪声复用；同时修改两个 Triton kernel 与两个公共函数签名。

```
# vllm/v1/worker/gpu/spec_decode/autoregressive/speculator.py (整理自变更后)
```

```
# _prefill 中初始化采样源位置：
# 位置 P 的输出隐藏态与 P+1 位置的 token 一起用于起草 P+2 的 token；
# 采样按“被采样 token 的前一个位置”取噪声，因此净调整量为 +1。
# sample_src_positions 独立于 input_buffers.positions 存在，且不会被
# max_model_len 截断，从根本上消除噪声复用。
sample_src_positions = positions + 1
# …… 运行草稿前向并取 sample_hidden_states ……
self.draft_tokens[:num_reqs, 0] = self.sample_draft(
    sample_hidden_states,
    sample_src_positions, # 显式传入，不再依赖 sample_draft 内部 +1
    idx_mapping,
    self.temperature,
    self.seeds,
    self.current_draft_step,
    self.draft_logits,
)
```

```
# _prepare_decode_inputs_kernel 中的推进逻辑：
# 每生成一个草案 token，就把采样流键推进 1。旧实现直接使用
# input_buffers.positions，该缓冲区接近 max_model_len 时会被截断，
# 导致相邻两步草案可能落在同一个噪声向量上（噪声复用）。
draft_token = tl.load(draft_tokens_ptr + req_idx * draft_tokens_stride)
tl.store(input_ids_ptr + req_idx, draft_token)
```

```
# Advance the draft sampling key.
sample_position = tl.load(sample_src_positions_ptr + req_idx)
tl.store(sample_src_positions_ptr + req_idx, sample_position + 1)
```

```
target_seq_len = tl.load(target_seq_lens_ptr + req_idx)
num_rejected = tl.load(num_rejected_ptr + req_idx)
seq_len = target_seq_len - num_rejected
```

tests/v1/spec_decode/test_rejection_sampler_utils.py

新增回归测试 `test_gumbel_drafted_rejection_sample_is_unbiased`，用窄词表高样本量复现生产草稿的噪声耦合，是证明修复有效的关键证据。

```
# tests/v1/spec_decode/test_rejection_sampler_utils.py (新增回归测试)
```

```
# 宽词表测试 (VOCAB_SIZE 个桶、10·VOCAB_SIZE 次试验、每桶约 10 个样本)
# 不足以分辨这种小幅分布偏差；16 个桶 + 20 万次试验、每桶约 1.2 万样本，
# 才能分辨几个百分点的偏差。
NARROW_VOCAB_SIZE = 16
NARROW_NUM_TRIALS = 200_000
```

```
def _gumbel_drafted_tokens(inputs, draft_logits_1d, num_trials, num_speculative_steps):
    """用 gumbel_sample 生成草案，使其与验证/残差重采样共用同一 (seed, pos) 流。

    _build_rejection_sample_inputs 原本用 torch.multinomial 抽草案，
    与重采样噪声天然独立；生产环境的草案来自 gumbel_sample，按
    pos[t*(K+1)+i] 取键，与 _rejection_kernel/_resample_kernel 读的是
    同一条噪声流。此辅助函数复现该耦合，用于暴露共享噪声导致的偏差。
    """

    k = num_speculative_steps
    vocab_size = draft_logits_1d.shape[0]
    device = draft_logits_1d.device
    draft_tokens = gumbel_sample(
        draft_logits_1d.unsqueeze(0).expand(num_trials * k, vocab_size).float(),
        inputs["expanded_idx_mapping"].view(num_trials, k + 1)[: , :k].reshape(-1).contiguous(),
        inputs["temperature"],
        inputs["seed"],
        inputs["pos"].view(num_trials, k + 1)[: , :k].reshape(-1).contiguous(),
        apply_temperature=True,
        is_drafting=True, # 关键：与生产草稿路径保持一致（加盐）
    )
    draft_sampled = torch.zeros(num_trials * (k + 1), dtype=torch.int64, device=device)
    draft_sampled.view(num_trials, k + 1)[: , 1:] = draft_tokens.view(num_trials, k)
    return draft_sampled
```

```
def test_gumbel_drafted_rejection_sample_is_unbiased(num_speculative_steps):
    """草案提案与残差重采样不得共享噪声向量。
```

若共享噪声：给定提案赢得 argmax，其他 token 的 Gumbel 都被截断在该最大值之下，草案排名最高的 token 截断最紧，于是残差低估的正是这些 token，输出分布不再等于目标分布。敏感性对照：去掉 _gumbel_drafted_tokens 里的 is_drafting=True 后，position 0 的 chi2 从约 12 暴涨到约 1500（阈值约 70），而宽词表测试仍然通过——这正是必须用窄词表的原因。

```
"""
# 反向排序的草案 (draft = -target) 会拒绝约 73% 的提案，
# 大多数试验都会走到残差重采样，偏差信号得以放大。
target_logits_1d = torch.randn(NARROW_VOCAB_SIZE, device="cuda", dtype=torch.float32)
draft_logits_1d = -target_logits_1d
inputs = _build_rejection_sample_inputs(
    target_logits_1d, draft_logits_1d, num_speculative_steps,
    temperature=1.0, num_trials=NARROW_NUM_TRIALS,
)
inputs["draft_sampled"] = _gumbel_drafted_tokens(
    inputs, draft_logits_1d, NARROW_NUM_TRIALS, num_speculative_steps
)
sampled, num_sampled = rejection_sample(**inputs, num_speculative_steps=num_speculative_steps)
# position 0 承载全部检验力（每个试验都到达），后续只有被接受才到达
assert (num_sampled >= 1).all()
```

```

target_probs = torch.softmax(target_logits_1d, dim=0)
for pos in range(num_speculative_steps + 1):
    accepted_mask = num_sampled >= pos + 1
    _assert_distribution_match(
        sampled[accepted_mask, pos], target_probs, "cuda", label=f"position {pos}"
    )

```

评论区精华

本 PR 来自 fork, [claude\[bot\]](#) 自动审核被跳过 ("automated review is disabled"), 维护者 [WoosukKwon](#) 直接 APPROVE 并回复 "Thank you!", 没有逐行 review 评论。真正的技术论证都沉淀在 PR body 和测试设计里, 核心论点包括: 验证是概率比检验 $p(x) > u \cdot q(x)$ 而非 Gumbel 耦合, 接受率的期望只依赖 $x \sim q$, 因此解耦噪声流理论上不应改变接受率——2M 次独立试验的固定 p/q 探针实测 0.1068 (基线) vs 0.1071 (本 PR) 支持该结论; 以及宽词表测试每桶仅约 10 个样本、无法分辨小幅分布偏差, 必须用窄词表高样本量测试才能暴露 (chi2 约 12 vs 约 1500 的对照)。

- fork PR 自动审核跳过, 维护者直接批准 (other): 批准合并; 技术正确性由作者的分布化测试与基准背书。

风险与影响

- 风险: 风险点如下: (1) RNG 行为变更: 所有 `draft_sample_method="probabilistic"` 的输出 token 序列会与修复前不同 (这是修复本意), 依赖旧有偏采样的外部系统需要重新评估; (2) 跨模块约定重构: 位置约定横跨 autoregressive、MTP、dflash、dflash2、dspark 五套 speculator, 任何一处的 +1 偏移偏差都会静默引入新的噪声流错位或碰撞, 现有测试覆盖了分布无偏性与流分离, 但未见基于真实模型 logits 的端到端统计验证; (3) CUDA graph 路径改动: `sample_src_positions` 的推进逻辑落在 `_prepare_decode_inputs_kernel` 与 `_update_draft_inputs_kernel` 两个 Triton 内核, 且 `prepare_decode_inputs/update_draft_inputs` 签名变更若遗漏调用点会直接崩溃 (本 PR 内已同步全部调用点); (4) 盐的碰撞安全依赖“位置远小于 2^{30} ”的前提, int64 位置下成立, 未来若位置语义变化需重新论证。
- 影响: 用户侧: 概率投机采样 (`draft_sample_method="probabilistic"`) 的输出分布恢复为目标分布, 属于正确性修复而非性能改动; 默认 greedy 完全不受影响。系统侧: V1 所有 speculator 的采样键统一为“各站点自报约定”, 新增 speculator 时必须显式处理 `sample_src_positions` 与 `is_drafting`, 契约更清晰。团队侧: 噪声流与位置约定的注释文档大幅改善, 5 个投机模型系列 (MTP、DFlash、DSpark 等) 的采样正确性有了可维护的统一基线; 基准数据为后续性能对比提供了参考锚点。
- 风险标记: 核心采样路径变更, 跨 5 个 speculator 重构, RNG 流语义变化, CUDA graph 捕获路径改动, 模型级统计测试覆盖有限

关联脉络

- PR #47386 (PR 正文引用的上游工作, 标题未在上下文中提供): PR body 明确说明 "Picking up the work from #47386", 本 PR 是其中 Gumbel 噪声耦合问题的延续与最终修

复。

- PR #53790 [Bugfix] NemotronHMTP: add hf_to_vllm_mapper so quant exclusions reach the MTP draft: 同为投机解码 + MTP 链路修复, 覆盖 multi_module_mtp speculator 周边逻辑, 与本 PR 的 MTP 采样位置改动同属一个功能域。
- PR #51705 [ROCm][MLA][DCP] Support causal multi-token verification: 涉及投机解码的目标验证语义 (多 token 验证), 与本 PR 的验证 / 重采样噪声流正确性同属验证链路演进。
- PR #53574 [Bugfix][SM120] DSv4: pass contiguous C128A decode topk indices on SM120: DeepSeek V4 投机解码采样路径的稳定性修复, 与本 PR 的 DSpark 采样侧改动同域。