

PR #51367 完整报告

vllm-project/vllm

[Kernel] Use Murmur3 RNG for Gumbel sampling

合并时间: 2026-09-16 14:52

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51367>

执行摘要

- 一句话: Gumbel 采样改用 Murmur3, 修复尾部概率偏差
- 推荐动作: 值得精读, 尤其是三处「跨后端数值可移植性」的设计: murmur3_uniform32 的 split-16 精确构造 (用两个可精确表示的半部绕开后端 uint32→fp32 转换)、murmur3_uniform64 用 domain=0x9E3779B9 拼 53 位并显式打开右端点、_log1p_neg_stable 的分段多项式替代 libdevice log1p。阅读时建议重点验证四点: $h = (\text{seed} \wedge \text{seed})$ 与 $h \wedge 16$ 是否为有意为之; $u \rightarrow 1$ 退化分支的噪声下限是否可接受; 删除入口测试的「交错完成」断言是否应由别的测试补位; 性能数字是否会在最终 head 上复现。对维护者的动作建议: 在 release note 中显式提示 seeded 非 greedy 输出序列变更, 并补一条 $u \rightarrow 1$ 端点的数值回归测试。

功能与动机

PR body 明确写了动机: Replace the Gumbel sampler's Philox-based uniform generation . . The goal is to preserve the target sampling distribution while improving RNG resolution and avoiding backend-specific math paths, 并关联 RFC #54143, 同时区分于保留 Philox、只多取输出位的 #47524。作者在评论中进一步给出量化证据: 24 位截断路径在 vocab=129,280、logit gap 18/20 时尾部质量 z 分数为 -5.89 / -5.16, 改用完整 32 位源后收敛到 -1.18 / -1.09, 确认截断就是偏差主因; 此外 Ascend A3 上 Philox 采样 kernel 比 Murmur3 慢 26-89 倍, 属于典型的后端专属数学路径成本。

实现拆解

1. 入口换 RNG 并重排噪声流隔离: vllm/v1/worker/gpu/sample/gumbel.py 的 gumbel_noised_argmax 删除 gumbel_seed = tl.randint(seed, pos) 与 tl_rand32/tl_rand64 调用, 改为按 (seed, pos, keys) 直接调用 murmur3_uniform32/64; draft 与 target 的隔离从「偏移 offset」改为「对 pos 加 _DRAFT_NOISE_SALT」, 因为 seed 与 pos 现在都直接进入 hash, salt 位置即可保证两条噪声流不一致。同时移除 tldevice 导入 (不再需要后端 log1p), _TL_RAND_MIN 的注释改写为仅供 rejection sampling 使用。
2. 新增 Murmur3 哈希族: _murmur3_rotl32 (无内置 rotate, 用移位或拼回)、_murmur3_mix (body: 乘 0xCC9E2D51、rotl 15、乘 0x1B873593、异或、rotl 13、乘 5 加 0xE6546B64)、_murmur3_fm32 (finalizer) 以及对外入口 murmur3_hash32(seed, pos, offset, domain=0)。murmur3_hash32 先把 64 位 seed

拆成高低 32 位混合、再混 pos、最后混词表内 offset，并用 domain 常量在同一坐标上再开一条独立流。

- 两种均匀数构造：murmur3_uniform32 用「拆 16 位再转 fp32」的方式实现 $(\text{random32} + 0.5) * 2^{32}$ ，两个半部都能被 fp32 精确表示，因此不依赖后端 uint32→fp32 转换也能用满 32 位分辨率；murmur3_uniform64 用 domain=0x9E3779B9 取第二个 hash，拼出 53 位有效位后交给 _uniform64_from_random53，并用 $\text{tl.where}(\text{uniform} == 1.0, \text{uniform} - 1.1102230246251565e-16, \text{uniform})$ 显式打开右端点，避免最大中点舍入到 1.0 后 $\log(1-u)$ 变成 -inf。
- fp32 尾部对数可移植化：新增 _log1p_neg_stable，在 $u < 0.25$ （决定 argmax 胜出的大 Gumbel 噪声尾部）走 8 次 Horner 多项式（绝对误差 $< 6e-7$ ），其余区间退化为 $\text{tl.log}(\text{tl.maximum}(1 - u, 5.96e-08))$ ，用分段近似替代 libdevice log1p，从而避免后端专属数学库差异。
- 测试与配套改动：tests/v1/worker/test_gpu_gumbel_sample.py 新增最小 Triton kernel _murmur3_known_answer_kernel 与 test_murmur3_known_answers（锁定 6 组已知答案，含 int64 极值 seed 与 $1 < \text{pos} < 30$ 附近的 pos）、_uniform64_endpoint_kernel 与 test_uniform64_excludes_endpoints（断言结果为 2^{54} 与 $1 - 2^{53}$ ），并把 test_drafting_uses_a_separate_noise_stream 的 docstring 从「salts the Philox offset」更新为「salts the RNG position」。三处输出敏感的入口测试被去随机化：test_include_reasoning.py 四个用例加 temperature=0 并把 max_tokens 从 200 提到 512；test_struct_output_generate.py 中结构化输出用例 temperature=1.0 改 0；test_completion.py 删除 test_parallel_no_streaming 与 test_parallel_streaming 里「子请求必须不同长度」的 output_token_lengths / chunk_lengths 断言。

文件	类型	关键动作
vllm/v1/worker/gpu/sample/gumbel.py	源码	新增 Murmur3 RNG 族与分段 log1p，替换采样内核的 RNG 调用点
tests/v1/worker/test_gpu_gumbel_sample.py	测试	已知答案 + fp64 端点回归，更新噪声流语义的文档字符串
tests/entrypoints/openai/chat_completion/test_include_reasoning.py	测试	改为 temperature=0、max_tokens=512，去掉对采样流的依赖
tests/entrypoints/openai/completion/test_completion.py	测试	删除并行补全交错完成的长度断言
tests/entrypoints/llm/test_struct_output_generate.py	测试	结构化输出用例改贪心，避免 RNG 变更导致 flaky

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 _murmur3_rotl32, _murmur3_mix, _murmur3_fm32, murmur3_hash32）：唯一

源码改动文件，承载全部新 RNG 实现与采样内核调用点切换，决定采样分布正确性与跨后端行为

- tests/v1/worker/test_gpu_gumbel_sample.py (模块 采样测试; 类别 test; 类型 test-coverage; 符号 _murmur3_known_answer_kernel, _uniform64_endpoint_kernel, test_murmur3_known_answers, test_uniform64_excludes_endpoints) : 为新的 Murmur3 路径补已知答案与 fp64 端点回归, 并更新 draft 噪声流语义的断言说明, 是本次数值正确性的主要保护网
- tests/entrypoints/openai/completion/test_completion.py (模块 补全接口测试; 类别 test; 类型 test-coverage; 符号 test_parallel_no_streaming, test_parallel_streaming) : 并行补全测试删除「子请求交错完成」断言, 是本次唯一明确的能力回退, 直接反映采样流变更对测试结构的冲击
- tests/entrypoints/openai/chat_completion/test_include_reasoning.py (模块 推理内容测试; 类别 test; 类型 test-coverage; 符号 test_include_reasoning_true_non_streaming, test_include_reasoning_false_non_streaming, test_include_reasoning_true_streaming, test_include_reasoning_false_streaming) : 四个推理内容用例改为 temperature=0 并提高 max_tokens, 使结果不再依赖新的 RNG 流, 是入口层配套去随机化的代表
- tests/entrypoints/llm/test_struct_output_generate.py (模块 结构化输出; 类别 test; 类型 test-coverage; 符号 test_structured_output_batched_with_non_structured_outputs_requests) : 结构化输出与非结构化混批用例由 temperature=1.0 改 0, 避免 RNG 变更导致的 flaky, 是影响最小但必读的一行改动

关键符号: gumbel_noised_argmax, murmur3_hash32, murmur3_uniform32, murmur3_uniform64, _uniform64_from_random53, _log1p_neg_stable, _murmur3_mix, _murmur3_fm32, test_murmur3_known_answers, test_uniform64_excludes_endpoints

关键源码片段

tests/v1/worker/test_gpu_gumbel_sample.py

为新的 Murmur3 路径补已知答案与 fp64 端点回归, 并更新 draft 噪声流语义的断言说明, 是本次数值正确性的主要保护网

tests/v1/worker/test_gpu_gumbel_sample.py: 新增的已知答案与端点测试

@triton.jit

```
def _murmur3_known_answer_kernel(
    seeds_ptr, positions_ptr, offsets_ptr,
    hashes_ptr, domain_hashes_ptr,
    numel: tl.constexpr, BLOCK_SIZE: tl.constexpr,
):
    # 被测函数无法直接在 Python 侧调用, 这里用最小 kernel 把设备端真实实现
    # 搬出来, 保证测的是 Triton 里的 murmur3_hash32 而不是 Python 复刻
    idx = tl.arange(0, BLOCK_SIZE)
    mask = idx < numel
    seed = tl.load(seeds_ptr + idx, mask=mask)
```

```

pos = tl.load(positions_ptr + idx, mask=mask)
offset = tl.load(offsets_ptr + idx, mask=mask)
hash32 = murmur3_hash32(seed, pos, offset).to(tl.int64)
domain_hash32 = murmur3_hash32(seed, pos, offset, domain=0x9E3779B9).to(tl.int64)
tl.store(hashes_ptr + idx, hash32, mask=mask)
tl.store(domain_hashes_ptr + idx, domain_hash32, mask=mask)

```

```
def test_murmur3_known_answers():
```

```
    # 覆盖 0 / 1 / -1 / 跨 32 位 seed / int64 两端极值, pin 住跨后端 hash 一致性
```

```
    seeds = torch.tensor(
        [0, 1, -1, 0x123456789ABCDEF, -(1 << 63), (1 << 63) - 1],
        dtype=torch.int64, device=DEVICE,
    )
```

```
    positions = torch.tensor(
        [0, 2, 17, 123456789, 1 << 30, (1 << 30) + 7],
        dtype=torch.int64, device=DEVICE,
    )
```

```
    offsets = torch.tensor(
        [0, 3, 1023, 199999, 4096, 0xFFFFFFFF],
        dtype=torch.int64, device=DEVICE,
    )
```

```
    hashes = torch.empty_like(seeds)
    domain_hashes = torch.empty_like(seeds)
```

```
    _murmur3_known_answer_kernel[(1,)](
        seeds, positions, offsets, hashes, domain_hashes,
        numel=seeds.numel(), BLOCK_SIZE=8,
    )
```

```
    assert hashes.tolist() == [
        0x8134CDF8, 0x23B53F91, 0xC8C13C14,
        0x5DDE974F, 0xF1A9355B, 0xD952F9AA,
    ]
```

```
    # domain=0x9E3779B9 的独立流同样锁答案, 防止 fp64 路径悄悄退化
```

```
    assert domain_hashes.tolist() == [
        0xD28AD962, 0xE3AAF818, 0xD223C51C,
        0x0547D2FE, 0x3AED160F, 0x49E8EA7F,
    ]
```

```
def test_uniform64_excludes_endpoints():
```

```
    # random53 = 0 与  $2^{53} - 1$  对应最靠近两端的样本: 期望  $2^{54}$  与
```

```
    #  $1 - 2^{53}$ , 即区间必须保持开区间, 否则  $\log(1 - u)$  会出现  $-\inf$ 
```

```
    random53 = torch.tensor([0, (1 << 53) - 1], dtype=torch.int64, device=DEVICE)
```

```
    uniform = torch.empty(2, dtype=torch.float64, device=DEVICE)
```

```
    _uniform64_endpoint_kernel[(1,)](
        random53, uniform, numel=random53.numel(), BLOCK_SIZE=2,
    )
```

)

```
assert torch.isfinite(uniform).all()
assert ((uniform > 0.0) & (uniform < 1.0)).all()
assert uniform.tolist() == [2.0**-54, 1.0 - 2.0**-53]
```

评论区精华

讨论区 30 条里绝大多数是 CI/bot 编排 (/ci run、mergify 冲突提示、pre-commit 失败提示)，真正的技术交锋集中在作者自证的几条长评论上：

作者 freyfwf: Changed fp32 Murmur uniform from 24-bit truncation to full 32-bit source resolution: $u = (\text{random32} + 0.5) / 2^{32}$, reconstructed through two exact 16-bit int→fp32 conversions for backend portability. Verified split-16 reconstruction against correctly rounded fp32 on 2M random uint32 values: 0 mismatches.

作者 freyfwf: For vocab=129,280 and 100k draws, tail-mass z-scores were -1.18 at logit gap 18 and -1.09 at gap 20. The old 24-bit path gave -5.89 and -5.16 respectively, confirming the truncation was the material bias source.

作者 freyfwf (性能结论克制) : The host was shared with an unrelated 8-GPU workload ... so these results are used as a no-material-regression check, not as a clean speedup claim.

作者 freyfwf (自陈覆盖缺口) : the available runtime image predates the is_drafting call-site update. Both variants therefore used the same one-line is_drafting=False harness adapter ... drafting/speculative behavior is outside its scope.

评审侧只有两条机器 / 单人信号：claude[bot] 在 fork 场景下先说明自动评审被禁用，随后给出「Code review found no issues / No high-confidence issues detected in this change」，Isotr0py 给出 APPROVED。review_comments_count 为 0，意味着没有逐行 diff 讨论、没有张量统计之外的第三方复核，PR 的正当性几乎完全建立在作者自测数据上。

- fp32 均匀数从 24 位截断改为完整 32 位源分辨率 (correctness): 确认 24 位截断是偏差主因并已修复；旧路径在 far-tail 上系统性低估，新路径回到统计噪声范围内。
- fp64 均匀数端点必须保持开区间 (correctness): 端点已闭合为开区间并有回归测试锁定；同时避免依赖 Triton 可能按 fp32 物化的近似 1 字面量。
- draft / target 噪声流隔离方式从 offset salt 改为 pos salt (design): 隔离语义保持等价，依赖「pos 为 int64 且远不接近 2^{30} 」的既有假设；spec-decode 验证的概率比语义未变。
- 输出敏感的入口测试去随机化与「交错完成」断言删除 (testing): 去 flaky 的目标达成，但 HTTP/ 流式层的交错完成覆盖净减少；引擎层单测与入口层时序验证不等价，这一取舍没有第三方复核。
- 自动化评审与人工 approval 的实际深度 (question): 缺少逐行复核与独立数值复算，合入依据主要是作者自测数据与 CI 绿灯 (Buildkite #86309 与后续重跑)。

风险与影响

• 风险:

1. 确定性输出变更 (兼容性): body 已声明 seeded 非 greedy 请求会产生不同的确定性 token 序列; 作者实测 base/head 只有 1/16 首轮补全完全一致。任何依赖 seed 复现的输出基线 (用户侧回归脚本、内部 golden 输出、缓存 / 重放对比) 都会失效, 但 PR 声明 No documentation change is needed, 对外可见的行为变更缺少 release note。
2. RNG 统计强度的论证方式: Murmur3 是非密码学哈希, 作者用 Pearson z 分数 (mean -0.009, std 0.998, max |z| 3.90)、byte/bit/serial 检验与 Monte Carlo 尾部质量来论证, 但这些是经验检验, 不等价于 Philox 这类 counter-based PRNG 的已知 equidistribution 性质。另外 murmur3_hash32 里 $h = (\text{seed} \wedge \text{seed}).\text{to}(\text{tl.uint32})$ 恒为 0 (疑似「清零并转 uint32」的遗留写法), $\text{_murmur3_fmix32}(h \wedge 16)$ 把 16 当长度字节, 两处都偏离标准 MurmurHash3 写法, 需要确认是刻意设计还是可读性陷阱。
3. fp32 对数近似的边界: _log1p_neg_stable 只在 $\text{value} < 0.25$ 用多项式, 其余走 $\text{tl.log}(\text{tl.maximum}(1 - \text{value}, 5.96\text{e-}08))$; 当 u 接近 1 时 fp32 相减本身损失精度并被 clamp, 退化分支给出的噪声下限约为 $\log(5.96\text{e-}08) \approx -16.7$, 属于对下尾的显式取舍, 但目前没有测试直接覆盖 $u \rightarrow 1$ 端。
4. 测试覆盖的净回退: test_completion.py 删除的「子请求在不同长度完成」断言是唯一在 HTTP/ 流式层观察交错完成的位置, 作者以 ParentRequest 单元测试已确定性覆盖为由删除; 单请求引擎层测试与端到端流式时序不是同一层, 覆盖确实变薄。
5. 性能结论的证据强度: Ascend A3 的 26-89 倍加速微基准早于最终 head (作者自述 predate the final fp32-uniform, logarithm, and fp64-endpoint changes), 模型级 A/B 又跑在与 8 卡无关负载共享的主机上 (输出吞吐 CV 11.5%/7.4%), 且 drafting 路径未端到端覆盖。当前 head 的 kernel 延迟尚未复测, 性能收益不能直接作为既成事实引用。
6. 安全面: seed 由调用方提供、Murmur3 可逆性远高于 Philox, 理论上降低了「seed 泄露即可预测采样」的门槛, 但采样 RNG 本身不承担安全职责, 风险等级低。- 影响: 影响面集中在 Model Runner V2 的非 greedy 采样路径: 所有使用 seed 的 temperature > 0 请求在升级后得到不同 (但分布等价) 的输出, 可复现性基线与下游 golden 对比需要重建; 未设置 seed 的用户无感知。收益侧, 非 NVIDIA 后端 (Ascend A3 采样 kernel 从 160 ms 级降到 1.8 ms 级) 与结构化输出 / 推理解析等多请求场景最明显, RTX 5090 上模型级 A/B 显示输出吞吐 +4.15%、mean TPOT -5.15%, 可读作无实质回归。工程侧影响包括: draft/target 噪声流隔离方式从 offset salt 改为 pos salt, spec-decode 的验证语义保持不变但依赖「pos 永不接近 2^{**30} 」的既有假设; 入口测试 (推理内容、结构化输出、并行补全) 因改贪心或删除断言而更稳定, 但代价是少了一层对采样流的端到端约束。greedy 路径、logits cache 行为、公开 API 与 kernel launch 结构均不变, 说明爆炸半径被有意限制在单个采样文件。- 风险标记: 确定性输出变更, 非密码学哈希充当 RNG, 测试覆盖部分回退, 性能数据待最终 head 复测, drafting 路径未端到端覆盖

关联脉络

- PR #55055 [BugFix][Model Runner V2][Spec Decode] Fix decode instance's multi-layer MTP kv caches during P/D: 同属 Model Runner V2 worker 与 spec-decode

路径；本 PR 的 draft/target 噪声流 salt 语义直接作用于 MTP/ 投机解码的采样一致性。

- PR #57112 [ROCm][CI] Fix MLA RoPE fused-kernel tests for TheRock image: 同样是融合内核在不同后端上的数值容差问题，采用按 ULP 分级容差的测试策略，可与本 PR 用多项式近似控制 $6e-7$ 误差的取舍对照阅读。
- PR #56096 [XPU] Route to fused_qk_rmsnorm_rope_gate triton kernel: 同类「用可移植 Triton 内核替换后端专属路径换取显著加速」的演进范式，本 PR 在 Ascend 上的 26-89 倍收益与其动机一致。
- PR #56904 [Bugfix] Make GPU sync checks safe under torch.compile: 同样涉及 Triton/ 编译环境下工具的兼容与测试稳定性处理，与本 PR 去 flaky 的测试改造思路相近。