PR #46701 完整报告

vllm-project/vllm

[Core][V1] Support trace_decode_token_ids for deterministic decode replay

合并时间：2026-08-20 15:12

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46701

执行摘要

- 一句话：新增 trace_decode_token_ids 实现确定性解码重放
- 推荐动作：值得精读。重点关注：采样后注入再算 logprob 的顺序设计、以 GPU 状态推导 step 避免 CPU 同步、UVA StagedWriteTensor 的批量写入模式、以及"配置开关 + 输入归一化 + 统一校验"三层防护。对想扩展 MRV2 sampler 或实现类似重放 / 调试功能的工程师是很好的参考。合并前建议在 GPU 上跑通 examples/generate/trace_replay_offline.py 端到端验证。

功能与动机

PR body 明确其目标：让请求跟随固定 token trace 并返回模型对该 trace 的真实 logprobs，用于"Replaying the same trace under different configs (dtype, quantization, attention/MoE backend, TP size, etc.) ... apples-to-apples measure of numerical divergence"；以及 RL 训练中"recompute logprobs for a rollout's exact tokens under the training engine, so the rollout↔train logprob gap can be measured and minimized"。同时解释为何必须走 decode 路径而非 prefill 打分：许多模型 prefill 与 decode 使用不同的算子或内核，只有逐 token 走 decode 路径才能忠实复现解码分布。

实现拆解

1. 采样参数与统一校验：vllm/sampling_params.py 新增 trace_decode_token_ids: list[int] | None 字段，并在 from_optional/clone 中透传。_validate_trace_replay 在 verify() 中统一完成非空、n==1、非负整数、vocab 范围及投机解码冲突等检查，避免注入越界 token。
2. 配置开关与 V2 限制：vllm/config/vllm.py 与 vllm/config/model.py 增加 enable_trace_replay（CLI --enable-trace-replay），_verify_trace_replay_config 要求 Model Runner V2；vllm/engine/arg_utils.py 接入参数，未开启时请求被拒绝。
3. 输入归一化：vllm/v1/engine/input_processor.py 的 _normalize_trace_replay_params 将 max_tokens 覆盖为 trace 长度与剩余上下文的最小值、强制 ignore_eos、清空 eos_token_id/stop_token_ids/stop 字符串，并按剩余上下文截断 trace，防止写穿 max_model_len 宽的 GPU 缓冲。
4. GPU 执行内核：vllm/v1/worker/gpu/sample/trace_replay.py 新增 TraceReplayState（UVA 后备 StagedWriteTensor）与 Triton kernel _trace_replay_kernel；sampler.py 在采样和 logprob 计算之间调用 apply_trace，以 total_len - prompt_len 推导 step，原地覆盖 sampled，logprobs 仍来自真实分布。apply_staged_writes 与 any_trace 门控保证无

trace 请求零额外开销。

5. 测试与示例：新增 3 个测试文件覆盖字段校验、输入归一化与内核行为；examples/generate/trace_replay_offline.py 演示先贪婪生成再逐 token 重放并断言一致。

关键文件：

- vllm/v1/worker/gpu/sample/trace_replay.py（模块 采样器；类别 source；类型 core-logic；符号 TraceReplayState, init, add_request, apply_staged_writes）：核心实现：TraceReplayState 管理每请求 trace 缓冲，Triton kernel 在采样后、logprob 计算前原地覆盖 sampled token，是功能正确性的关键。
- vllm/sampling_params.py（模块 采样参数；类别 source；类型 core-logic；符号 _validate_trace_replay）：定义新字段与统一校验入口，拦截空列表、n>1、非法 / 越界 token 及投机解码等不兼容组合，是请求准入的第一道防线。
- vllm/v1/engine/input_processor.py（模块 输入处理；类别 source；类型 core-logic；符号 _normalize_trace_replay_params）：归一化 trace 语义：覆盖 max_tokens、强制 ignore_eos、清除 EOS/stop 条件并按剩余上下文截断 trace，防止提前终止与 GPU 写穿。
- vllm/v1/worker/gpu/sample/sampler.py（模块 采样器；类别 source；类型 dependency-wiring；符号 Sampler.init, Sampler.call）：在采样热路径挂接 TraceReplayState.apply_trace，保证注入发生在采样之后、logprob 计算之前，是 logprobs 保持真实的关键接线点。
- vllm/config/vllm.py（模块 引擎配置；类别 source；类型 configuration；符号 _verify_trace_replay_config）：引擎配置开关与 V2 runner 限制，决定功能是否启用及显存是否分配。
- examples/generate/trace_replay_offline.py（模块 示例脚本；类别 source；类型 entrypoint；符号 build_llm, run_normal_generation, run_trace_replay, run_demo）：官方示例：先贪婪生成获取 token 序列，再以 trace 重放并逐 token 校验与打印 logprob，展示端到端用法。
- tests/v1/worker/test_gpu_trace_replay.py（模块 内核测试；类别 test；类型 test-coverage；符号 _i32, _i64, _trace_state, _set_lens）：直接覆盖 Triton kernel 行为：trace 覆盖、越界不动、非 trace 不动、idx_mapping 间接索引与负值跳过。
- tests/v1/engine/test_input_processor_trace_replay.py（模块 引擎测试；类别 test；类型 test-coverage；符号 _make_request, _normalize, test_normalize_trace_replay_params, test_trace_longer_than_remaining_context_is_truncated）：覆盖输入归一化与准入门控：max_tokens 覆盖、EOS/stop 清理、trace 截断、--enable-trace-replay 开关与 V2 runner 限制。
- tests/v1/sample/test_trace_replay_params.py（模块 参数测试；类别 test；类型 test-coverage；符号 test_sampling_params_trace_field_defaults_to_none, test_sampling_params_trace_field_accepts_list, test_sampling_params_trace_field_preserved_by_clone, test_sampling_params_trace_field_rejects_empty_list）：覆盖 SamplingParams 字段行为：默认值、clone 保留、空 / 非法 / 越界 token 拒绝与 n>1 拒绝。

关键符号：_validate_trace_replay, _normalize_trace_replay_params, TraceReplayState.add_request, TraceReplayState.apply_staged_writes, TraceReplayState.apply_trace, apply_trace_tokens, _trace_replay_kernel, _verify_trace_replay_config

关键源码片段

vllm/sampling_params.py

定义新字段与统一校验入口，拦截空列表、n>1、非法 / 越界 token 及投机解码等不兼容组合，是请求准入的第一道防线。

```
# vllm/sampling_params.py 中的校验逻辑片段
def _validate_trace_replay(
    self,
    model_config: ModelConfig,
    speculative_config: SpeculativeConfig | None,
) -> None:
    # 统一校验 trace replay 的请求级兼容性
    trace = self.trace_decode_token_ids
    if trace is None:
        return

    # 空 trace 没有意义
    if len(trace) == 0:
        raise ValueError('trace_decode_token_ids must be a non-empty list.')

    # trace 是逐请求单序列，n > 1 时 kernel 的 idx_mapping 长度对不上
    if self.n != 1:
        raise ValueError('trace_decode_token_ids requires n=1.')

    # 非法 token id（负数、非 int）会在注入后被当作 gather 索引
    if not all(isinstance(t, int) and t >= 0 for t in trace):
        raise ValueError(
            'trace_decode_token_ids must contain non-negative integers.'
        )

    # 超出 vocab 的 token 会越过 [batch, vocab_size] logprobs 的张量边界，
    # 造成 GPU 越界读并可能把内存内容作为 logprob 返回给用户
    vocab_size = model_config.get_vocab_size()
    out_of_vocab = [t for t in trace if t >= vocab_size]
    if out_of_vocab:
        raise VLLMValidationError(
            'trace_decode_token_ids contains out-of-vocab token ids: '
            + str(out_of_vocab)
        )

    # 投机解码的 draft 路径会改写 sampled，与强制注入冲突
    if speculative_config is not None:
```

```
            raise ValueError(
                'trace_decode_token_ids is not supported with speculative '
                'decoding.'
            )
```

vllm/v1/engine/input_processor.py

归一化 trace 语义：覆盖 max_tokens、强制 ignore_eos、清除 EOS/stop 条件并按剩余上下文截断 trace，防止提前终止与 GPU 写穿。

```
# vllm/v1/engine/input_processor.py 中的归一化逻辑片段
@staticmethod
def _normalize_trace_replay_params(
    processor: InputProcessor,
    sampling_params: SamplingParams,
    prompt_len: int,
) -> None:
    # 把 trace replay 的生成语义应用到请求级参数上，必须在
    # update_from_generation_config 之后调用，以便覆盖其缓存的 EOS 状态
    trace = sampling_params.trace_decode_token_ids
    if trace is None:
        return

    # trace 不能超出剩余上下文，否则会写穿 max_model_len 宽的 GPU 行
    max_tokens = min(
        len(trace),
        processor.model_config.max_model_len - prompt_len,
    )
    sampling_params.max_tokens = max_tokens
    sampling_params.min_tokens = 0

    # trace 内部的 EOS 或 stop 不代表生成提前结束，必须全部清除
    sampling_params.ignore_eos = True
    sampling_params.eos_token_id = None
    sampling_params.stop_token_ids = []
    sampling_params.stop = None
    sampling_params.all_stop_token_ids = set()

    # 截断后的 trace 才是实际要重放的序列，与 max_tokens 保持一致
    sampling_params.trace_decode_token_ids = trace[:max_tokens]
```

评论区精华

- njhill 在首轮 review 明确只能加到 MRV2，因为 MRV1 即将废弃；最终提交专门删除 MRV1 实现并拒绝 MRV1 请求。
- depthfirst-app 报告 HIGH 级越界读：未校验 vocab_size 时，trace token 被用作 torch.gather 索引，可越界读 GPU 内存并回传给多租户调用者；随后在 _validate_trace_replay 增加 vocab 校验。

- Codex 与 depthfirst-app 分别指出 stop 字符串未清理、trace 超出 max_model_len 会写穿 GPU 缓冲、condense() 未裁剪 trace 列表造成无界增长、prompt_logprobs 下 kernel 覆盖错误行，均在后续提交逐一修复。
- njhill 认为为 niche 功能分配 max_num_reqs x max_model_len 大张量不合理，要求加配置开关；新增 --enable-trace-replay 并按需分配。
- 关于不兼容组合的处理，最终由“fallback with warning”改为直接拒绝：njhill 表示“Requests should fail rather than silently falling back”。
- out-of-vocab token 导致 GPU 越界读 (security): 在 SamplingParams._validate_trace_replay 增加 vocab 范围校验，测试覆盖 out-of-vocab 拒绝。
- 仅支持 Model Runner V2，移除 MRV1 实现 (design): 最终提交删除 MRV1 实现，MRV1 下请求被拒绝并提示需要 V2。
- trace 与 stop 条件冲突导致提前终止 (correctness): 归一化时清空 stop 字符串、EOS 与 stop token，并强制 ignore_eos。
- condense() 未裁剪 trace 列表造成无界增长 (correctness): 在 InputBatch.condense() 中补充 trace_decode_token_ids 的裁剪。
- 大显存缓冲需要配置门控 (performance): 新增 --enable-trace-replay 开关，未开启时不分配缓冲且拒绝请求。
- prompt_logprobs 下 kernel 覆盖错误行 (correctness): 拒绝 trace 与 prompt_logprobs 组合使用。
- 不兼容组合 fallback 还是拒绝 (design): 改为直接拒绝，并更新示例文档说明所有被拒组合。

风险与影响

- 风险：
  - 核心采样路径每步引入一次 Triton kernel 启动；若 any_trace 门控或 sticky 状态在请求槽位复用或回收时未正确重置，可能误注入历史 trace。
  - _normalize_trace_replay_params 与 TraceReplayState.add_request 都对 trace 长度有隐含假设（不超过 max_model_len），两处逻辑必须保持同步，否则会写穿 GPU 缓冲。
  - InputBatch.condense() 已补裁剪 trace 列表，但这类平行维护的字段（类似 spec_token_ids）在后续新增字段时容易遗漏。
  - 越界读风险虽已通过 vocab 校验缓解，但校验集中在 SamplingParams.verify()，若存在绕过 verify 的内部调用路径，仍有隐患。
  - MRV2-only 策略意味着 MRV1 用户无法使用；TraceReplayState 依赖 RequestState 的 total_len/prompt_len 内部布局，随 MRV2 重构可能漂移。
- 影响：
  - 用户侧：为调试、RL 与数值对比提供官方、确定性的 logprob 重算能力；默认关闭，普通推理无感知。
  - 系统侧：开启开关后每请求预留 max_num_reqs x max_model_len x 4B 显存（UVA 缓冲）；关闭时零额外显存。

- 团队侧：统一了“指定 token 序列取 logprob”的入口，替代各团队自建补丁；SamplingParams 中调试 /RL 参数分组也为后续参数提供了范式。
- 风险标记：核心采样路径每步注入，GPU 越界读 / 写风险（已缓解），大显存缓冲依赖配置开关，MRV2-only 限制，与多特性组合需显式拒绝

关联脉络

- PR #53017 [Model Runner V2][Spec Decode] Fix draft logits cache column stride in gumbel_sample: 同为 Model Runner V2 采样路径的修复，与本 PR 共享 vllm/v1/worker/gpu/sample 目录与测试风格，反映 MRV2 采样层正在快速演进。
- PR #52839 [refactor] consolidate cp attn ops: 本 PR 在合并 main 时多次与 v1 worker 重构（如 ThinkingBudgetState 与 TraceReplayState 共存于 sampler.py）发生冲突，52839 同期也在重构 v1 attention 算子目录，二者同属 V1/MRV2 基础设施演进脉络。
- PR #52998 [Distributed] Enable FlashInfer all-reduce by default: 同周期 V1 执行链路的默认行为调整，与 trace-replay 一样影响 v1 生产路径的默认语义，可对照观察 V1 行为变更的评审标准。