

PR #52242 完整报告

vllm-project/vllm

[Feature][DSpark]: Logprobs adaptive verification

合并时间: 2026-08-25 22:22

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/52242>

执行摘要

- 一句话: DSpark 自适应校验解锁 logprobs 功能
- 推荐动作: 值得精读, 重点看三处设计决策: 1) 为何不能类型重载而必须新增互斥字段 (EngineCoreOutputs 解码器基于注解构建, 存在联合类型限制); 2) 设备端边界如何搭既有异步 D2H 拷贝的便车并在 tolists() 中物化, 避免额外同步点; 3) 单 chunk 断言如何把自适应校验的压缩布局约束显式化。可对照 tests/v1/test_outputs.py 的新增测试理解契约边界。

功能与动机

PR body 明确说明目的是 "Lifts the restriction blocking logprobs with enable_adaptive_verification implemented in the EODC on #47808", 对应 [ssu#51873](#)。

动机根源: 自适应校验在设备端按请求分配草稿数量并压缩 logits, 而 CPU 侧 cu_num_logits_np 只是压缩前的均匀占位布局, 若继续拿它当 cu_num_generated_tokens 使用, 会把 logprob 行错配到其他请求; 因此旧实现直接在 SamplingParams._validate_spec_decode() 中抛 ValueError ("Output logprobs are not supported with DSpark confidence-based verification."), 把 logprobs 与自适应校验做成互斥。本 PR 移除该限制, 并让真实边界始终在设备端随 logits 一起流动。

实现拆解

1. 解除入口限制: 在 vllm/sampling_params.py 的 SamplingParams._validate_spec_decode() 中删除「enable_adaptive_verification 且设置 num_logprobs 即抛 ValueError」的分支, 使 logprobs 管线在自适应校验下首次可达。
2. 设备端边界接管: vllm/v1/worker/gpu/spec_decode/rejection_sampler.py 的 RejectionSampler.init() 保存 enable_adaptive_verification 配置; _get_logprobs_tensors() 在行数展开 (num_logits != num_reqs) 时, 自适应校验场景改传 cu_num_logits.clone() 设备 tensor (而非 cu_num_logits_np.tolist()), 因为 cu_num_logits_np 是压缩前的占位布局, 用它切分会错配 logprob 行; 同时 _verify_in_chunks() 对自适应校验分支加断言, 保证压缩后的批次必然落在单 chunk, 不走依赖 CPU 布局的多 chunk 重建路径。
3. 契约放宽: vllm/v1/worker/gpu/sample/logprob.py 的 compute_topk_scores() 参数类型由 list[int] | None 放宽为 list[int] | torch.Tensor | None, 并按是否 tensor 分流写入 LogprobsTensors 的两个互斥字段 (cu_num_generated_tokens 或新增的

cu_num_generated_tokens_tensor) 。

4. 异步回传与物化：vllm/v1/outputs.py 的 LogprobsTensors 新增第 5 个字段 cu_num_generated_tokens_tensor；to_cpu_nonblocking() 把该 tensor 与其他 logprobs 张量同流以 non_blocking=True 排队 D2H 拷贝；tolists() 在拷贝事件后调用 .tolist() 物化为列表；filter() 与 cat() 增加互斥断言，防止该字段走不支持的多 chunk 路径。
5. 涟漪修复与配套：新增字段使所有 4 元解包失效，逐一修复 vllm/v1/engine/logprobs.py、vllm/v1/worker/gpu_model_runner.py（改为 *_通配解包）；同步调整 tests/v1/engine/test_output_processor.py、tests/v1/worker/test_gpu_rejection_sampler_chunking.py；tests/v1/test_outputs.py 新增 test_logprobs_tensors_tolist_with_tensor_boundaries 覆盖 tensor 边界经 to_cpu_nonblocking().tolist() 交接后 slice_request 切分正确；docs/features/speculative_decoding/adaptive_verification.md 同步移除限制说明。

关键文件：

- vllm/v1/outputs.py（模块 输出处理；类别 source；类型 core-logic；符号 LogprobsTensors.tolist, LogprobsTensors.to_cpu_nonblocking, LogprobsTensors.filter, LogprobsTensors.cat）：LogprobsTensors 新增 cu_num_generated_tokens_tensor 字段，并贯通 to_cpu_nonblocking() 的非阻塞 D2H 回传与 tolists() 的物化逻辑，是本次数据契约的核心载体。
- vllm/v1/worker/gpu/spec_decode/rejection_sampler.py（模块 拒绝采样；类别 source；类型 core-logic；符号 RejectionSampler.init, RejectionSampler._get_logprobs_tensors, RejectionSampler._verify_in_chunks）：自适应校验开启时改传设备端 cu_num_logits 作为每请求边界，并新增单 chunk 断言，是本次功能解锁的采样侧关键改动。
- vllm/sampling_params.py（模块 采样参数；类别 source；类型 core-logic；符号 SamplingParams._validate_spec_decode）：删除阻止 logprobs 与自适应校验并用的 ValueError 分支，是本次功能解锁的入口。
- vllm/v1/worker/gpu/sample/logprob.py（模块 概率计算；类别 source；类型 core-logic；符号 compute_topk_scores）：compute_topk_scores() 参数类型放宽并分流写入两个边界字段，是数据契约的中间转换点。
- tests/v1/test_outputs.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_logprobs_tensors_tolist_with_tensor_boundaries）：新增 test_logprobs_tensors_tolist_with_tensor_boundaries，覆盖 tensor 边界经异步拷贝与 tolists 后 slice_request 切分正确。
- vllm/v1/engine/logprobs.py（模块 对数管理；类别 source；类型 core-logic；符号 _update_prompt_logprobs）：修复 LogprobsTensors 新增字段导致的 4 元解包失效，改为通配解包。
- vllm/v1/worker/gpu_model_runner.py（模块 执行编排；类别 source；类型 data-contract）：同步修复 LogprobsTensors 解包，兼容新增 tensor 边界字段。
- tests/v1/engine/test_output_processor.py（模块 处理器测试；类别 test；类型 test-coverage）：测试配套调整，适配 LogprobsTensors 新字段。
- tests/v1/worker/test_gpu_rejection_sampler_chunking.py（模块 分块测试；类别 test；类型 test-coverage）：为拒绝采样 chunking 测试补充自适应校验断言相关配置。

- docs/features/speculative_decoding/adaptive_verification.md (模块 功能文档; 类别 docs; 类型 documentation) : 同步移除文档中 logprobs 不支持自适应校验的说明。

关键符号: RejectionSampler._get_logprobs_tensors, RejectionSampler._verify_in_chunks, compute_topk_scores, LogprobsTensors.tolist, LogprobsTensors.to_cpu_nonblocking, LogprobsTensors.cat, LogprobsTensors.filter, SamplingParams._validate_spec_decode

关键源码片段

vllm/v1/outputs.py

LogprobsTensors 新增 cu_num_generated_tokens_tensor 字段, 并贯通 to_cpu_nonblocking() 的非阻塞 D2H 回传与 tolists() 的物化逻辑, 是本次数据契约的核心载体。

```
class LogprobsTensors(NamedTuple):
    # [num_reqs x num_generated_tokens, max_num_logprobs + 1]
    logprob_token_ids: torch.Tensor
    # [num_reqs x num_generated_tokens, max_num_logprobs + 1]
    logprobs: torch.Tensor
    # [num_reqs x num_generated_tokens]
    selected_token_ranks: torch.Tensor
    # [num_reqs + 1]
    cu_num_generated_tokens: list[int] | None = None
    # [num_reqs + 1]; 自适应校验下真实边界只在设备端存在,
    # 因此放在 tensor 字段里随异步 D2H 拷贝一起回传,
    # 此时 cu_num_generated_tokens 保持 None, 两者互斥。
    cu_num_generated_tokens_tensor: torch.Tensor | None = None

    def tolists(self, cu_num_generated_tokens: list[int] | None = None):
        # 设备端边界在此物化为 list, 之后 slice_request 等
        # CPU 侧逻辑才能按真实 per-request 边界切分 logprob 行。
        if cu_num_generated_tokens is None:
            if self.cu_num_generated_tokens_tensor is not None:
                cu_num_generated_tokens = self.cu_num_generated_tokens_tensor.tolist()
            else:
                cu_num_generated_tokens = self.cu_num_generated_tokens
        return LogprobsLists(
            self.logprob_token_ids.cpu().numpy(),
            self.logprobs.cpu().numpy(),
            self.selected_token_ranks.cpu().numpy(),
            cu_num_generated_tokens,
        )

    def to_cpu_nonblocking(self) -> "LogprobsTensors":
        # 张量已在 CPU 时直接复用, 避免多余的拷贝。
        if self.logprob_token_ids.device.type == "cpu":
            return self
        # 边界 tensor 与 logprobs 张量在同一流上排队 D2H 拷贝,
```

```

# 保证 tolists() 读取前数据已就绪，且拷贝保持非阻塞。
cu_tensor = self.cu_num_generated_tokens_tensor
if cu_tensor is not None:
    cu_tensor = cu_tensor.to("cpu", non_blocking=True)
return LogprobsTensors(
    self.logprob_token_ids.to("cpu", non_blocking=True),
    self.logprobs.to("cpu", non_blocking=True),
    self.selected_token_ranks.to("cpu", non_blocking=True),
    self.cu_num_generated_tokens,
    cu_tensor,
)

```

vllm/v1/worker/gpu/spec_decode/rejection_sampler.py

自适应校验开启时改传设备端 cu_num_logits 作为每请求边界，并新增单 chunk 断言，是本次功能解锁的采样侧关键改动。

```

def _get_logprobs_tensors(
    self,
    sampled: torch.Tensor,
    num_sampled: torch.Tensor,
    logits: torch.Tensor,
    cu_num_logits: torch.Tensor,
    cu_num_logits_np: np.ndarray,
    max_num_logprobs: int,
) -> LogprobsTensors | None:
    if max_num_logprobs == NO_LOGPROBS:
        return None

    num_reqs = cu_num_logits.shape[0] - 1
    num_logits = logits.shape[0]
    # 先把每个请求的采样 token 按真实边界展开成扁平数组，
    # 与 logits 的展平行一一对应。
    flat_sampled = torch.zeros(
        num_logits, dtype=sampled.dtype, device=sampled.device
    )
    _flatten_sampled_kernel([(num_reqs,)](
        flat_sampled,
        sampled,
        sampled.stride(0),
        num_sampled,
        cu_num_logits,
        num_warps=1,
    ))

    # 常规 spec decode 下，一行 logits 对应一个请求；
    # 自适应校验会在 forward 后压缩 logits，因此行数会大于请求数。
    expanded_logits = num_logits != num_reqs
    cu_num_generated_tokens: list[int] | torch.Tensor | None = None
    if expanded_logits:

```

```

    if self.enable_adaptive_verification:
        # 自适应校验只把真实边界保存在设备端 (cu_num_logits) ,
        # cu_num_logits_np 仍是压缩前的均匀占位布局, 不能用于切分。
        cu_num_generated_tokens = cu_num_logits.clone()
    else:
        cu_num_generated_tokens = cu_num_logits_np.tolist()
    return compute_topk_scores(
        logits,
        max_num_logprobs,
        flat_sampled,
        cu_num_generated_tokens,
        logits_mode=self.sampler.logprobs_mode
        in ("raw_logits", "processed_logits"),
    )

```

评论区精华

核心交锋集中在两处设计验证：

1) benchislett 建议直接重载 `cu_num_generated_tokens` 类型而非新增互斥字段，作者回应 `core_client.py` 启动时基于类型注解构建 `EngineCoreOutputs` 解码器，不支持 `list[int] | torch.Tensor | None` 联合类型 (`TypeError: Type unions containing a custom type may not contain any additional types other than None`)，因此必须新增字段，benchislett 接受并最终 APPROVED；2) benchislett 要求验证新增 tensor 的 D2H 拷贝确实非阻塞，作者用 worker 端 profiler 证明每步仅 36 字节 `Memcpy DtoH (Device -> Pinned)` 且与既有 `logprobs` 拷贝同流，`VLLM_GPU_SYNC_CHECK=error` 端到端跑通。

- 边界字段设计取舍：类型重载 vs 新增互斥字段 (design): 接受双字段方案：
`cu_num_generated_tokens` 保持 `list[int] | None`，新增
`cu_num_generated_tokens_tensor` 承载设备端边界；benchislett 回复 "huh, alright" 并最终 APPROVED。
- 边界 tensor 的非阻塞 D2H 拷贝验证 (performance): 验证通过，确认拷贝为异步且与 `logprobs` 张量同流排队，无隐式同步。

风险与影响

- 风险：
 1. 数据契约变更：LogprobsTensors 从 4 字段增至 5 字段，NamedTuple 解包不报错，残留的 4 元解包会静默错绑；本 PR 已修复 `vllm/v1/engine/logprobs.py` 与 `vllm/v1/worker/gpu_model_runner.py` 并改用 `*` 通配，但其他间接调用方仍需回归关注。
 2. 异步时序敏感：`tolists()` 依赖 D2H 拷贝事件完成后读取边界，若调用顺序变化（如在 `to_cpu_nonblocking` 之前 `tolist`）会引入隐式同步或读到未就绪数据；当前有 `sync checker` 与 `profiler` 佐证，但属时序敏感代码。
 3. chunk 假设耦合：rejection_sampler.py 的 `_verify_in_chunks()` 新断言依赖「自适应校验压缩后恒落单 chunk」的前提（由 `max_chunk_logits` 预算保证），未来调整分块策略

会导致断言触发。

4. 额外开销：每步多一次 `cu_num_logits.clone()` 与 36 字节 DtoH 拷贝，profiler 显示可忽略，但极短步长场景有微小 kernel 启动成本。
5. 测试覆盖面：新增的是单元测试与作者自述 E2E，仓库 CI 未添加自适应校验 + logprobs 的端到端用例。
 - 影响：用户侧：DSpark/dflash 自适应校验用户首次可获得正确 logprobs，logprob 行切分从均匀占位修正为真实 per-request 边界，采样结果与 logprobs 自洽（作者验证 2048/2048 token 在自身 logprobs 字典内、0/32 请求漂移）。系统侧：v1 输出管线的 LogprobsTensors 数据契约扩展，影响 prompt logprobs 与 decode logprobs 两条路径的构造 / 解包。性能侧：每步新增约 36 字节设备侧拷贝，影响可忽略。团队侧：为后续把「设备端元数据随 D2H 拷贝回传」的模式推广到其他 spec decode 功能提供了参考范式。整体影响范围中低，集中在 vllm/v1 内部少量文件。
 - 风险标记：核心路径变更，异步时序敏感，数据契约变更，缺少 E2E 回归用例

关联脉络

- PR #53435 Dflash2 load fix: 同属 DSpark/dflash 推断链路，修复加载回归并补充 spec_decode 测试，与本 PR 形成互为回归保护。
- PR #53407 [Bugfix][MRV2] Dispatch uniform decode to a padded FULL cudagraph: 同为 mrv2 下 GPU 执行路径的调度修正，与本 PR 覆盖的 rejection sampler 与 cudagraph 运行环境相关。
- PR #49636 [Model][MoE] DeepSeek-V4: add opt-in FlashInfer moe_ep expert backend: 同属 DeepSeek 系模型的推理链路优化（DSpark 为 DeepSeek 草稿模型），反映该功能线的持续投入方向。