PR #23994 完整报告

sgl-project/sglang

[spec decoding] supports step 0 in adaptive spec decoding (updating draft kv cache without draft decoding)

合并时间：2026-06-16 13:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23994

执行摘要

- 一句话：自适应推测解码支持 step=0 与 draft KV 缓存更新
- 推荐动作：建议所有使用自适应推测解码（--speculative-adaptive）的用户阅读此 PR，特别是 _build_trivial_verify_input 的实现和 SGLANG_SPEC_SKIP_ZERO_STEP_DRAFT_EXTEND 的语义。该 PR 设计上保留了 draft KV 更新路径作为默认行为，是一个谨慎的权衡。测试覆盖完整，可直接合并。

功能与动机

PR 标题和修改内容表明，在自适应推测解码中允许 step=0（即零步推测）是填补之前 TODO 的关键能力。通过在高 batch 下禁用 drafting 避免浪费计算，同时保持 draft KV cache 更新以支持后续快速恢复，从而提升系统吞吐并降低延迟波动。

实现拆解

1. 配置开放 step=0：在 adaptive_spec_params.py 的 DEFAULT_ADAPTIVE_CONFIG 中为 batch size 8、32、64 引入候选步 0，并将验证逻辑从 s>0 放宽为 s>=0。
2. 核心调度分支：在 eagle_worker_v2.py 的 forward_batch_generation 中，当 speculative_num_steps == 0 时调用新方法 _build_trivial_verify_input 构造不包含真正 draft 的 EagleVerifyInput，并依据环境变量 SGLANG_SPEC_SKIP_ZERO_STEP_DRAFT_EXTEND 决定是否跳过 draft_extend（直接调用 _stub_skipped_draft_extend 模拟 stub 操作）。
3. EMA 更新保护：在 adaptive_spec_params.py 的 update 方法中，仅当 current_steps > 0 时才更新 EMA 统计；_recompute_params 增加从 step=0 恢复时的探测逻辑（直接跳至最小正步对应的 target）。
4. 后端创建适配：在 draft_utils.py 的 create_decode_backend 中将条件从 ==1 改为 <=1，避免为 step=0 创建 multi-step 后端。
5. 环境变量门控：在 environ.py 中新增 SGLANG_SPEC_SKIP_ZERO_STEP_DRAFT_EXTEND（默认关闭），用于控制是否跳过 draft_extend 以节省计算代价。
6. 集成与单元测试：新增 TestAdaptiveZeroStepBatchSizeServer 验证 server 在 batch 升降时正确切换 step（3→0→3）；新增 test_zero_step_mixed_slot_drops_probes_and_rechecks 验证 EMA 在 step=0 和 step>0 间切换的正确性；移除旧的

test_zero_steps_raises。

关键文件：

- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测解码；类别 source；类型 core-logic；符号 _build_trivial_verify_input, _stub_skipped_draft_extend）：核心调度逻辑变更，新增两个关键私有方法实现 step=0 的 verify input 构建和 draft_extend 跳过 stubbing。
- python/sglang/srt/speculative/adaptive_spec_params.py（模块 推测解码；类别 source；类型 core-logic；符号 _apply_target_steps）：默认配置和自适应算法核心逻辑修改，支持 step=0 并调整 EMA 更新条件。
- test/registered/spec/eagle/test_adaptive_speculative.py（模块 集成测试；类别 test；类型 test-coverage；符号 TestAdaptiveZeroStepBatchSizeServer, setUpClass, tearDownClass, _steps）：验证 step=0 在集成层正确工作，覆盖 server 真实环境。
- test/registered/unit/spec/test_adaptive_spec_params.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_zero_step_mixed_slot_drops_probes_and_rechecks, test_zero_steps_raises）：验证参数层在 step=0 时的切换逻辑和 EMA 行为。
- python/sglang/srt/environ.py（模块 环境变量；类别 source；类型 configuration）：新增环境变量门控，允许跳过 zero-step draft_extend。
- python/sglang/srt/speculative/draft_utils.py（模块 推测解码；类别 source；类型 core-logic）：避免为 step=0 创建 multi-step draft 后端。

关键符号：_build_trivial_verify_input, _stub_skipped_draft_extend, _apply_target_steps, _recompute_params, create_decode_backend

关键源码片段

python/sglang/srt/speculative/eagle_worker_v2.py

核心调度逻辑变更，新增两个关键私有方法实现 step=0 的 verify input 构建和 draft_extend 跳过 stubbing。

```
# python/sglang/srt/speculative/eagle_worker_v2.py
# 新增方法：构建一个仅含根 token 的 verify input，使得 TARGET_VERIFY 图直接接受并采样一个 bonus token

def _build_trivial_verify_input(self, batch: ScheduleBatch) -> EagleVerifyInput:
    """Build a 1-node EagleVerifyInput rooted at the previous bonus token.

    Used when ``speculative_num_steps == 0`` to skip drafting while still
    routing through the existing TARGET_VERIFY graph captured at
    ``draft_token_num=1``: the kernel always accepts the root and samples
    one new bonus token from target logits -- functionally a plain decode.
    """
    if batch.forward_mode.is_idle():
        return EagleVerifyInput.create_idle_input(
            topk=self.topk, spec_steps=0, num_verify_tokens=1
        )
```

```
    draft_input: EagleDraftInput = batch.spec_info
    bs = batch.seq_lens.shape[0]
    device = self.device

    # 构造检索索引和占位 next_token 数组
    retrieve_index = torch.arange(bs, dtype=torch.long, device=device).unsqueeze(1)
    retrieve_next_token = torch.full((bs, 1), -1, dtype=torch.long, device=device)
    retrieve_next_sibling = torch.full((bs, 1), -1, dtype=torch.long, device=device)

    attn_backend = self._target_worker.model_runner.attn_backend
    mask_buf, position_buf = attn_backend.get_verify_buffers_to_fill_after_draft()
    if mask_buf is not None:
        custom_mask = mask_buf
        custom_mask.fill_(True)
    else:
        if batch.seq_lens_sum is not None:
            seq_lens_sum = batch.seq_lens_sum
        elif batch.seq_lens_cpu is not None:
            seq_lens_sum = int(batch.seq_lens_cpu.sum())
        else:
            seq_lens_sum = bs * attn_backend.max_context_len
        custom_mask = torch.ones(seq_lens_sum + bs, dtype=torch.bool, device=device)

    if position_buf is not None:
        positions = position_buf
        positions[:bs].copy_(batch.seq_lens)
    else:
        # 没有 position buffer，手动构造
        positions = batch.seq_lens.to(torch.int32, copy=True)
        if batch.extend_lens is not None and batch.extend_lens.sum() > 0:
            positions += batch.extend_lens.to(device=device, dtype=torch.int32)

    return EagleVerifyInput(
        retrieve_index=retrieve_index,
        retrieve_next_token=retrieve_next_token,
        retrieve_next_sibling=retrieve_next_sibling,
        tree_mask=custom_mask,
        positions=positions,
        hidden_states=draft_input.hidden_states,
    )

# 新增方法：当跳过 draft_extend 时 stub 操作，更新 seq_lens 和重置 speculative logprob

def _stub_skipped_draft_extend(self,
                               batch: ScheduleBatch,
                               batch_output: Batch) -> None:
    """Stub the draft_extend that would normally follow verify.

    The draft KV cache is NOT updated: it will be stale until the batch shrinks
```

```
    and steps become positive again. However, we still need to advance seq_lens
    and reset speculative_logprob for correctness."""
    bs = batch.seq_lens.shape[0]
    new_len = batch.seq_lens + 1
    batch.seq_lens.copy_(new_len)
    # 重置 speculative_logprob 后，下次调用 draft_extend 时重新计算
    if hasattr(batch, 'speculative_logprob'):
        batch.speculative_logprob.fill_(0.0)

# forward_batch_generation 中新增 step==0 分支（关键控制流）
if self.speculative_num_steps == 0:
    # Drafting disabled (high batch size). _draft_extend below still
    # runs, keeping draft KV warm for when the batch shrinks.
    verify_input = self._build_trivial_verify_input(batch)
else:
    with ... # 原有 draft 上下文
        verify_input: EagleVerifyInput = self.draft_worker.draft(batch)
...
# 后续 verify 和 publish 逻辑保持不变
if (
    self.speculative_num_steps == 0
    and envs.SGLANG_SPEC_SKIP_ZERO_STEP_DRAFT_EXTEND.get()
):
    self._stub_skipped_draft_extend(batch, batch_output)
else:
    with ... # 原有 draft_extend 上下文
        self.draft_worker._draft_extend_for_decode(batch, batch_output)
```

python/sglang/srt/speculative/adaptive_spec_params.py

默认配置和自适应算法核心逻辑修改，支持 step=0 并调整 EMA 更新条件。

```
# python/sglang/srt/speculative/adaptive_spec_params.py
# 默认配置扩展：高 batch 时允许 step=0

DEFAULT_ADAPTIVE_CONFIG: dict[str, dict] = {
    "1": {"candidate_steps": [1, 3, 7], ...},
    "8": {"candidate_steps": [0, 1, 3], ...}, # 新增 0
    "32": {"candidate_steps": [0, 1], ...}, # 新增 0
    "64": {"candidate_steps": [0], ...}, # 新增 slot
}

# 在 _recompute_params 中，当 old_steps == 0 时直接跳到最小正步
if old_steps == 0:
    current_idx = min(current_idx + 1, len(self.candidate_steps) - 1)
    target = self.candidate_steps[current_idx]
    if target > 0 and self.ema_accept_len < 0:
        # 从无 draft 状态恢复时，为 target 设置中性 EMA
        self.ema_accept_len = float(target - 1)
    return self._apply_target_steps(old_steps, target)
```

```
# 在 drop 判断时对候选步 0 特殊处理：用 0.5 作为阈值
while current_idx > 0:
    prev_step = self.candidate_steps[current_idx - 1]
    drop_threshold = 0.5 if prev_step == 0 else prev_step - 0.5
    drop_threshold += self.down_hysteresis
    if self.ema_accept_len <= drop_threshold:
        current_idx -= 1
    else:
        break

# update 方法改为仅在 step>0 时更新 EMA
if self.current_steps > 0:
    batch_avg = sum(num_correct_drafts_per_req) / len(num_correct_drafts_per_req)
    self.ema_accept_len = (
        1 - self.ema_alpha
    ) * self.ema_accept_len + self.ema_alpha * batch_avg
```

评论区精华

该 PR 无 review 评论，仅获得 alphabetc1 的批准。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 核心路径变更：forward_batch_generation 是推测解码的主流程，新增分支可能影响正常验证路径，需确保 idle 状态和 mask/position buffer 正确处理。
  2. 环境变量复杂性：SGLANG_SPEC_SKIP_ZERO_STEP_DRAFT_EXTEND 引入一个新的调优维度，若用户错误启用可能导致 draft KV 严重过期，恢复后命中率骤降。
  3. EMA 逻辑调整：update 中条件跳过 EMA 更新可能导致 ema_accept_len 滞留在旧值，影响 step 切换决策（但测试已覆盖基本场景）。
  4. 兼容性：新的默认配置包含 step=0，若现有用户依赖旧配置且未指定自定义文件，可能意外启用 step=0；但 candidate_steps 列表允许 0 意味着 low batch 仍会使用非零步。- 影响：用户：在高并发场景下可借助自适应 step=0 自动禁用 draft，降低 GPU 内存和计算压力，提升公平性。普通用户无需额外配置即可受益（默认配置已覆盖）。系统：新增 _build_trivial_verify_input 在 step=0 时构造仅包含根 token 的 verify input，复用 TARGET_VERIFY 图，性能开销较低。团队：维护者需理解 step=0 的特殊逻辑和两个新的私有方法，但代码结构清晰，扩散风险低。
- 风险标记：核心调度路径变更，环境变量引入配置复杂性，EMA 逻辑条件更新可能影响步进

关联脉络

- 暂无明显关联 PR