# PR #7180 完整报告

PaddlePaddle/FastDeploy

[XPU] Unify Spec and non-spec branch.(#6947)

合并时间：2026-04-16 14:58

原文链接：http://prhub.com.cn/PaddlePaddle/FastDeploy/pull/7180

## 执行摘要

- 一句话：在 XPU 平台统一推测解码和非推测解码分支，新增草稿令牌验证算子。
- 推荐动作：该 PR 值得精读，重点关注 sampler 逻辑拆分、gather_next_token 接口统一以及 verify_draft_tokens 算子的设计，这些决策体现了跨平台架构对齐和模块化设计。

## 功能与动机

根据 PR body，动机是“【组网统一】 Unify Spec and non-spec branch.”和“Unify speculative decoding: align with GPU, refactor MTP flow.”，目的是统一分支并重构 MTP 流程以与 GPU 对齐。

## 实现拆解

1. 重构 Sampler 逻辑：在 fastdeploy/model_executor/layers/sample/sampler.py 中，将 forward_xpu 函数拆分为 _normal_sample_xpu（处理 NAIVE 模式）和 _verify_and_sample_xpu（处理验证模式），使用 verify_draft_tokens 算子进行草稿令牌验证。
2. 更新 gather_next_token 算子：修改 custom_ops/xpu_ops/src/ops/gather_next_token.cc，将 output_padding_offset 参数改为 is_speculative 布尔标志，统一接口并适配 speculative 解码路径。
3. 新增 verify_draft_tokens XPU 算子：在 custom_ops/xpu_ops/ 下新增 C++ wrapper 和 XPU3 kernel，实现 TOPP、GREEDY 和 TARGET_MATCH 三种验证策略，用于 draft token 验证。
4. 统一 share_inputs 字段：在 fastdeploy/worker/xpu_model_runner.py 和 fastdeploy/worker/input_batch.py 中，移除 XPU 特有的 output_cum_offsets 和 output_padding_offset，改用 cu_seqlens_q_output 和 batch_id_per_token_output，与 GPU 保持一致。
5. 更新测试配套：调整 custom_ops/xpu_ops/test/test_adjust_batch_and_gather_next_token.py 以适配新接口，并新增 test_verify_draft_tokens.py 进行验证算子测试。

关键文件：

- fastdeploy/model_executor/layers/sample/sampler.py（模块 采样器；类别 source；类型 core-logic；符号 forward_xpu, _normal_sample_xpu, _verify_and_sample_xpu）：核心采样逻辑重构，统一 spec 和 non-spec 分支，拆分为 naive 和验证采样路径。

- custom_ops/xpu_ops/src/ops/mtp/verify_draft_token.cc（模块 XPU 算子；类别 source；类型 core-logic）：新增 verify_draft_tokens 算子，实现草稿令牌验证，支持三种验证策略。
- fastdeploy/worker/xpu_model_runner.py（模块 模型运行器；类别 infra；类型 infrastructure）：统一 spec_method 命名和 proposer 初始化逻辑，更新 share_inputs 字段以对齐 GPU。

关键符号：forward_xpu, _normal_sample_xpu, _verify_and_sample_xpu, verify_draft_tokens, GatherNextToken

## 关键源码片段

### fastdeploy/model_executor/layers/sample/sampler.py

核心采样逻辑重构，统一 spec 和 non-spec 分支，拆分为 naive 和验证采样路径。

```
def _verify_and_sample_xpu(self, logits: paddle.Tensor, probs: paddle.Tensor,
                           sampling_metadata: SamplingMetadata, max_model_len: int,
                           share_inputs: Dict[str, paddle.Tensor], #
                           类型更正为Dict，实际按字符串键访问
                           accept_all_drafts: bool = False,
                           reject_all_drafts: bool = False) -> SamplerOutput:
    """Verify draft tokens (MTP/Ngram mode) on XPU using verify_draft_tokens."""
    from fastdeploy.model_executor.ops.xpu import top_p_candidates, verify_draft_tokens

    # 根据验证策略准备目标令牌
    if self.verify_strategy == VerifyStrategy.TARGET_MATCH:
        # 使用top-k/top-p采样生成目标令牌，与其他硬件路径对齐
        top_p, top_k, topp_seed = padding_sampling_params(
            sampling_metadata.top_p, sampling_metadata.top_k, sampling_metadata.seed,
            paddle.reshape(share_inputs["seq_lens_this_time"], shape=[-1]),
            paddle.reshape(share_inputs["seq_lens_encoder"], shape=[-1])
        )
        _, target_tokens = top_k_top_p_sampling(probs, top_p=top_p, top_k=top_k,
                                                top_k_list=sampling_metadata.top_k_list,
                                                topp_seed=topp_seed)
    elif self.verify_strategy == VerifyStrategy.GREEDY:
        # 贪心策略直接取概率最大令牌
        target_tokens = paddle.argmax(probs, axis=-1)
    elif self.verify_strategy == VerifyStrategy.TOPP:
        # 生成top-p候选列表，用于后续验证
        candidate_scores, candidate_ids, candidate_lens = top_p_candidates(
            probs, sampling_metadata.top_p, share_inputs["batch_id_per_token_output"],
            self.speculative_max_candidate_len, max_model_len
        )
    else:
        raise ValueError(f"Unknown verify strategy: {self.verify_strategy}")

    # 调用verify_draft_tokens算子进行验证，返回接受令牌数等信息
```

```
    verify_draft_tokens(...) # 参数省略，实际调用包含多个张量
    # 构建并返回采样输出对象
    return SamplerOutput(sampled_token_ids=share_inputs["accept_tokens"],
                         logprobs_tensors=None,
                         token_num_per_batch=share_inputs["accept_num"],
                         logits=logits)
```

## custom_ops/xpu_ops/src/ops/mtp/verify_draft_token.cc

新增 verify_draft_tokens 算子，实现草稿令牌验证，支持三种验证策略。

```
std::vector<paddle::Tensor> VerifyDraftTokens(
    const paddle::Tensor& step_output_ids,
    const paddle::Tensor& step_lens_this_time,
    const paddle::Tensor& seq_lens_encoder,
    const paddle::Tensor& seq_lens_decoder,
    const paddle::Tensor& target_tokens,
    const paddle::Tensor& candidate_ids,
    const paddle::Tensor& candidate_scores,
    const paddle::Tensor& candidate_lens,
    const paddle::Tensor& cu_seqlens_q_output,
    const paddle::Tensor& reasoning_status,
    const paddle::Tensor& max_dec_len,
    const paddle::Tensor& step_idx,
    int verify_strategy,
    int max_candidate_len) {
    // 初始化随机数状态，当前种子硬编码为0，可能影响TOPP采样多样性
    int random_seed = 0;
    std::vector<float> dev_curand_states_cpu(real_bsz);
    std::mt19937_64 rng(random_seed);
    for (int i = 0; i < real_bsz; ++i) {
        dev_curand_states_cpu[i] = rng();
    }

    // 根据上下文标志分配设备内存，存在RAII作用域问题需注意
    float* dev_curand_states_xpu = nullptr;
    if (xpu_ctx_flag) {
        xpu::ctx_guard RAII_GUARD(ctx);
        dev_curand_states_xpu = RAII_GUARD.alloc<float>(dev_curand_states_cpu.size());
        // 内存拷贝到XPU设备
    } else {
        // CPU路径直接使用主机内存指针
        dev_curand_states_xpu = dev_curand_states_cpu.data();
    }

    // 调用底层plugin实现验证逻辑
    fastdeploy::plugin::verify_draft_tokens(
        ctx, step_output_ids.data<int>(), step_lens_this_time.data<int>(),
        seq_lens_encoder.data<int>(), seq_lens_decoder.data<int>(),
        target_tokens.data<int>(), candidate_ids.data<int>(),
```

```
        candidate_scores.data<float>(), candidate_lens.data<int>(),
        cu_seqlens_q_output.data<int>(), reasoning_status.data<int>(),
        max_dec_len.data<int64_t>(), step_idx.data<int64_t>(),
        dev_curand_states_xpu, verify_strategy, max_candidate_len
    );

    return {step_output_ids}; // 返回更新后的输出张量
}
```

## 评论区精华

review 评论中，主要讨论点包括：WRAPPER_CHECK_PTR 类型参数错误（可能引发运行时问题）、随机种子硬编码为 0（影响 TOPP 采样多样性）、未使用的死代码（如 ClusterReduce 函数）、变量命名不一致（spec_method vs speculative_method）、reasoning_status 字段未更新（功能待实现）。决策上，维护者回复“后续会加上相关功能”，表明部分问题被接受但暂未解决。

- WRAPPER_CHECK_PTR 类型参数错误 (correctness): 未明确解决，建议修正类型参数。
- 随机种子硬编码影响采样多样性 (correctness): 建议从外部传入随机种子或使用动态种子，但未解决。
- 变量命名不一致 (style): 建议改回或添加注释说明，但未解决。
- reasoning_status 字段未更新 (design): 功能待实现，状态暂挂起。

## 风险与影响

- 风险：技术风险包括：WRAPPER_CHECK_PTR 类型错误（在 custom_ops/xpu_ops/src/plugin/src/wrapper/mtp_wrapper/verify_draft_tokens.cpp 中）可能导致内存访问问题或崩溃；随机种子固定（在 custom_ops/xpu_ops/src/ops/mtp/verify_draft_token.cc 中）使采样结果确定，可能影响输出多样性；XPU 不支持 NGRAM/SUFFIX 方法（在 fastdeploy/worker/xpu_model_runner.py 中），若配置可能导致运行时异常；reasoning_status 未更新（在 fastdeploy/worker/xpu_model_runner.py 中），相关推理功能可能失效。
- 影响：对用户：XPU 上的 speculative decoding 更统一，支持更多验证策略，提升使用体验。对系统：增强 XPU 后端功能，减少平台差异，便于维护。对团队：代码结构更清晰，但需注意兼容性和测试覆盖。
- 风险标记：类型检查错误，随机性固定，平台兼容性问题，未使用字段

## 关联脉络

- PR #6947 [XPU] add verify draft tokens: 直接相关，当前 PR 是 cherry-pick 或延续，新增 verify_draft_tokens 算子。
- PR #6685 未知（PR body 中提到）: 在 PR body 中引用，作为统一分支的原始 PR，可能涉及类似架构对齐。