PR #6947 完整报告

PaddlePaddle/FastDeploy

[XPU] add verify draft tokens

合并时间：2026-04-15 10:18

原文链接：http://prhub.com.cn/PaddlePaddle/FastDeploy/pull/6947

执行摘要

- 一句话：为 XPU 平台新增投机解码草稿令牌验证算子，支持三种验证策略。
- 推荐动作：建议精读此 PR，重点关注 XPU kernel 的实现细节（如验证策略逻辑和随机数处理），以及设计权衡（如线程安全修复）。对于从事投机解码或跨平台优化的工程师，此 PR 展示了硬件特定算子的集成模式，值得学习。

功能与动机

动机是将投机解码中的草稿令牌验证功能移植到 XPU 平台，以支持更高效的推测解码。PR body 中提到了关联 PR #6685，表明参考了已有的 GPU 实现；从 review 讨论可推断，旨在为 XPU 提供相同的验证能力，确保跨硬件一致性。

实现拆解

1. 新增 XPU kernel：在 custom_ops/xpu_ops/src/plugin/src/kernel/kunlun3cpp/mtp_kernel/verify_draft_tokens.xpu 中实现核心验证逻辑，支持 TOPP、GREEDY、TARGET_MATCH 三种策略，并包含 Phase1 验证和 Phase2 采样输出。
2. 添加 C++ wrapper 和主机端算子：在 verify_draft_tokens.cpp 和 verify_draft_token.cc 中包装 kernel 调用，处理输入输出张量、随机数生成（使用 std::atomic 修复线程安全）和错误检查。
3. 扩展 Python 绑定：修改 custom_ops/xpu_ops/src/ops/pybind/pybind.cc 暴露 verify_draft_tokens 算子到 Python 接口，便于上层调用。
4. 补充单元测试：在 custom_ops/xpu_ops/test/test_verify_draft_tokens.py 中添加大量测试用例，覆盖各种策略、验证窗口、边界条件（如 EOS 和最大长度），并包含参考实现对比。
5. 修复配套代码：修改 speculate_verify.cc 以支持随机种子持久化，确保与 GPU 实现生命周期对齐。

关键文件：

- custom_ops/xpu_ops/test/test_verify_draft_tokens.py（模块 测试模块；类别 test；类型 test-coverage；符号 to_paddle_inputs, run_kernel, run_ref, compare_results）：新增的单元测试文件，覆盖 verify_draft_tokens 算子的各种场景，是验证功能正确性的关键。
- custom_ops/xpu_ops/src/plugin/src/kernel/kunlun3cpp/mtp_kernel/verify_draft_tokens.xpu（模块 算子内核；类别 infra；类型 core-logic；符号 is_in_end, is_in, xorwow,

topp_sampling_kernel）：XPU kernel 核心实现文件，包含三种验证策略的逻辑和随机数采样，直接影响性能和行为正确性。

- custom_ops/xpu_ops/src/plugin/src/wrapper/mtp_wrapper/verify_draft_tokens.cpp（模块 包装层；类别 infra；类型 infrastructure）：C++ wrapper 文件，负责参数校验和 kernel 调用封装，是连接主机端和设备端的关键层。
- custom_ops/xpu_ops/src/ops/mtp/verify_draft_token.cc（模块 算子入口；类别 infra；类型 entrypoint；符号 VerifyDraftTokens）：主机端算子实现文件，处理 Paddle 张量输入输出，并管理随机数种子，是用户调用的入口点。
- custom_ops/xpu_ops/src/ops/pybind/pybind.cc（模块 Python 绑定；类别 infra；类型 configuration）：Python 绑定文件，暴露 verify_draft_tokens 算子到 Python 接口，便于上层集成和测试。

关键符号：verify_draft_tokens, VerifyDraftTokens, topp_sampling_kernel, is_in, is_in_end, verify_one_match, verify_one_topp

关键源码片段

custom_ops/xpu_ops/test/test_verify_draft_tokens.py

新增的单元测试文件，覆盖 verify_draft_tokens 算子的各种场景，是验证功能正确性的关键。

```
import numpy as np
import paddle
from fastdeploy.model_executor.ops.xpu import verify_draft_tokens
from fastdeploy.spec_decode import VerifyStrategy

# 设备选择：优先使用 XPU，否则回退到 CPU
DEVICE_PLACE = paddle.XPUPlace(0) if paddle.is_compiled_with_xpu() else paddle.CPUPlace()

def to_paddle_inputs(inputs: Dict[str, Any]) -> Dict[str, Any]:
    """将 numpy 输入字典转换为 Paddle 张量，并放置到设备上。"""
    paddle_inputs = {}
    for k, v in inputs.items():
        if isinstance(v, (int, bool, float, str)):
            paddle_inputs[k] = v # 标量类型直接传递
        elif v is not None:
            paddle_inputs[k] = paddle.to_tensor(v, place=DEVICE_PLACE) # 张量转换
        else:
            paddle_inputs[k] = None # 处理空值
    return paddle_inputs

def run_kernel(paddle_inputs: Dict[str, Any], inputs: Dict[str, Any]):
    """调用 verify_draft_tokens kernel，执行草稿令牌验证。"""
    verify_draft_tokens(
        paddle_inputs["step_output_ids"], # 输出令牌 ID
        paddle_inputs["step_output_len"], # 输出长度
        paddle_inputs["step_input_ids"], # 草稿令牌输入
        paddle_inputs["target_tokens"], # 目标模型输出
```

```
        paddle_inputs["candidate_ids"], # 候选 ID 集
        paddle_inputs["candidate_scores"], # 候选分数
        paddle_inputs["candidate_lens"], # 候选长度
        paddle_inputs["topp"], # top-p 参数
        paddle_inputs["stop_flags"], # 停止标志
        paddle_inputs["seq_lens_encoder"], # 编码器序列长度
        paddle_inputs["seq_lens_this_time"], # 当前序列长度
        paddle_inputs["end_tokens"], # 结束令牌
        paddle_inputs["is_block_step"], # 是否块步骤
        paddle_inputs["cu_seqlens_q_output"], # 累计序列长度
        paddle_inputs["reasoning_status"], # 推理状态
        paddle_inputs["max_dec_len"], # 最大解码长度
        paddle_inputs["step_idx"], # 步骤索引
        inputs["max_seq_len"], # 最大序列长度
        inputs["verify_window"], # 验证窗口
        inputs["verify_strategy"], # 验证策略（0=TOPP,1=GREEDY,2=TARGET_MATCH）
        inputs["reject_all"], # 是否拒绝所有
        inputs["accept_all"] # 是否接受所有
    )
```

custom_ops/xpu_ops/src/plugin/src/kernel/kunlun3cpp/mtp_kernel/verify_draft_tokens.xpu

XPU kernel 核心实现文件，包含三种验证策略的逻辑和随机数采样，直接影响性能和行为正确性。

```
namespace fd_xpu3 {
// 检查令牌是否在结束令牌列表中
__device__ bool is_in_end(const int64_t id,
                          __global_ptr__ const int64_t *end_ids,
                          int length) {
    for (int i = 0; i < length; i++) {
        if (id == end_ids[i]) {
            return true; // 命中结束令牌
        }
    }
    return false; // 未命中
}

// 检查草稿令牌是否在候选集中
__device__ inline bool is_in(__global_ptr__ const int64_t *candidates,
                             const int64_t draft,
                             const int candidate_len) {
    for (int i = 0; i < candidate_len; i++) {
        if (draft == candidates[i]) {
            return true; // 找到匹配
        }
    }
    return false; // 无匹配
}
```

```
// TOPP 采样 kernel：根据随机数和 top-p 参数选择令牌
__device__ int64_t topp_sampling_kernel(__global_ptr__ const int64_t *candidate_ids,
                                        __global_ptr__ const float *candidate_scores,
                                        __global_ptr__ const float *dev_curand_states,
                                        const int candidate_len,
                                        const float topp) {
    const int tid = core_id();
    float sum_scores = 0.0f;
    float rand_top_p = *dev_curand_states * topp; // 使用随机状态
    for (int i = 0; i < candidate_len; i++) {
        sum_scores += candidate_scores[i];
        if (rand_top_p <= sum_scores) {
            return candidate_ids[i]; // 返回选中的令牌 ID
        }
    }
    return candidate_ids[0]; // 默认返回第一个候选
}

// GREEDY/TARGET_MATCH 策略的单个令牌验证
__device__ inline bool verify_one_match(int64_t target_token,
                                        int64_t draft_token) {
    return target_token == draft_token; // 精确匹配
}

// TOPP 策略的单个令牌验证
__device__ inline bool verify_one_topp(__global_ptr__ const int64_t *verify_tokens_row,
                                       int64_t draft_token,
                                       int actual_cand_len) {
    return is_in(verify_tokens_row, draft_token, actual_cand_len); // 检查是否在候选集中
}
}
```

评论区精华

review 中核心讨论包括：

- 类型检查错误：fastdeploy-bot 指出 wrapper 中多个参数（如 seq_lens_this_time、cu_seqlens_q_output）类型检查错误，应为 int 但误写为 float 或 bool，可能导致运行时崩溃；作者已修复。
- 线程安全问题：fastdeploy-bot 和 Copilot 强调全局静态变量 g_seed 和 g_offset 在多线程环境下存在竞态条件，建议使用 std::atomic；作者回应“已修复”并改为原子操作。
- 随机数生成：讨论中提及随机数种子固定为 0 可能导致重复序列，以及 curand_states 布局不一致问题；后续提交修复了种子生成逻辑。
- 代码风格建议：如变量命名混淆（CUDA_PLACE 实际是 XPUPlace）、未使用函数 xorwow，作者在提交历史中修复了部分问题。
- 类型检查错误 (correctness): 作者在后续提交中修复了类型检查，使用正确的 WRAPPER_CHECK_PTR 调用。

- 线程安全问题 (design): 作者回应“已修复”，在提交历史中将变量改为 std::atomic 以确保原子操作。
- 随机数生成风险 (correctness): 提交历史中多次修复（如“fix ramdom seed”），改进了种子生成逻辑和状态管理。

风险与影响

- 风险：技术风险包括：
  1. 线程安全风险：尽管已使用 std::atomic，但全局状态在多线程并发调用时仍可能引入微妙竞态，影响随机数序列的独立性。
  2. 类型检查错误：wrapper 中参数类型校验错误若未完全修复，可能导致内存访问越界或未定义行为。
  3. 随机数逻辑不一致：curand_states 在 CPU 和 XPU kernel 中的索引方式不一致，可能引发越界或验证结果偏差。
  4. 性能风险：新增 XPU kernel 未经充分性能测试，在高负载下可能影响投机解码的整体吞吐。
  5. 兼容性风险：与 GPU 实现（PR #6685）的行为对齐度需确保，否则可能引入跨平台不一致问题。
- 影响：影响范围：
- 用户：XPU 平台的投机解码功能得到增强，用户可利用新算子提升令牌验证效率，可能改善推理速度。
- 系统：新增算子和测试增加了代码库复杂性，但全面的单元测试有助于维护稳定性；需确保 XPU 环境编译和运行正常。
- 团队：为跨硬件开发提供了参考实例，但需要持续监控线程安全和随机数生成的一致性，以避免回归。
- 风险标记：线程安全问题，类型检查错误，随机数生成风险，性能未充分验证

关联脉络

- PR #6685 未知：PR body 中提到的关联 PR，可能为 GPU 平台上的 verify_draft_tokens 实现，本 PR 旨在将其移植到 XPU。
- PR #7323 [Speculative Decoding] Support mtp super ultra overlap in pd-split mode with insert_task overlap: 同为投机解码相关优化，涉及 MTP 超重叠和性能提升，技术领域重叠。
- PR #7071 [XPU] support glm-4.5-air (fix neox+partial_rotary_factor): 同为 XPU 平台功能扩展，涉及模型支持和算子优化，可参考跨硬件实现模式。