PR #35072 完整报告

sgl-project/sglang

[Intel XPU] support prefill only models for xpu

合并时间：2026-08-24 16:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35072

执行摘要

- 一句话：为 Intel XPU 补齐 prefill-only 模型测试覆盖
- 推荐动作：值得精读。推荐关注三点：一是 cross-encoder 数值敏感性的技术权衡——为何 bf16 下融合内核的归约顺序差异会导致超出 1e-2 容差，这是选择 attention backend 与 dtype 的关键依据；二是 decoder-only rerank 的分数提取方法（yes/no token logprob 转概率比）；三是 classification 测试中 " 概率比较优于原始 logits 比较 " 的稳定性设计，以及 review 中显存双占问题的修正过程。

功能与动机

PR body 明确说明目的是为 Intel XPU 设备增加 prefill-only 模型（embedding/rerank/classification/reward）的测试用例，并给出 5 类模型的 dtype 与 attention backend 组合表。Issue 评论中 gaopengff 进一步解释：cross-encoder 因使用双向注意力（causal=False）并输出原始 logit，对累加误差非常敏感；融合内核（intel_xpu、flashinfer）的归约顺序与 eager 参考不一致，在 bf16 下会超出 1e-2 容差，因此需要对齐 CUDA 参考测试（test/registered/prefill_only）的 triton + fp32 配置。

实现拆解

本 PR 以测试补齐为主线，不涉及生产代码路径改动，具体拆解如下：

1. 重构 XPU embedding 测试（test/registered/xpu/test_xpu_embedding.py）：将原先基于 OpenAI 兼容接口的端到端 server 测试改为 HF/SRT 输出对比的 parity 测试。新增 _truncate_prompts 按 max_position_embeddings 截断超长输入，新增 assert_close_prefill_logits 以余弦相似度校验 embed logits，并新增 test_matryoshka_embedding 验证 128 维 matryoshka embedding。
2. 新增 XPU rerank 测试（test/registered/xpu/test_xpu_rerank.py）：分两类覆盖。Decoder-only 风格（Qwen/Qwen3-Reranker-0.6B）构造 yes/no 二元 token 的 logprob 并计算 P(yes)/(P(yes)+P(no)) 作为分数，使用 intel_xpu backend + bfloat16；Cross-encoder 风格（BAAI/bge-reranker-v2-m3）因数值敏感性使用 triton backend + float32，并设置 chunked_prefill_size=-1、disable_radix_cache=True 对齐 CUDA 参考测试。
3. 新增 XPU 分类测试（test/registered/xpu/test_xpu_classification.py）：对 jason9693/Qwen2.5-1.5B-apeach 分别用 HFRunner（cross_encoder 模式）和

SRTRunner（embedding 模式读 embed_logits 作为 class logits）得到 softmax 概率，比较形状、max_abs_diff 与 top class 一致性，容差为 5e-2。

4. 新增 XPU reward 测试（test/registered/xpu/test_xpu_reward.py）：用 HFRunner 与 SRTRunner 分别计算 Skywork-Reward-V2-Qwen3-0.6B 的 reward 分数并做逐元素比较，容差为 1.5e-1；SRT 侧通过 apply_chat_template 将对话转为 prompts。
5. 配套与 review 改进：所有测试均调用 register_xpu_ci 注册到 stage-b-test-1-gpu-xpu suite，est_time 设为 60-180 秒不等。评审阶段已按 mingfeima 的反馈，将 classification 测试调整为顺序执行 HFRunner/SRTRunner，避免同一块 XPU 上双模型同时占用显存。

关键文件：

- test/registered/xpu/test_xpu_rerank.py（模块 重排序；类别 test；类型 test-coverage；符号 format_prompt, yes_no_token_ids, score_from_token_logprobs, TestXPUDecoderRerank）：本次最核心的交付物，同时覆盖 decoder-only 与 cross-encoder 两种 rerank 风格，分数提取与容差设计最具技术含量。
- test/registered/xpu/test_xpu_classification.py（模块 分类任务；类别 test；类型 test-coverage；符号 TestXPUClassification, _hf_probs, _srt_probs, test_classification_logits）：新增分类任务测试；review 讨论集中在显存双占与概率容差设计上，最终改为顺序执行 HFRunner/SRTRunner，是测试设计调整的直接体现。
- test/registered/xpu/test_xpu_embedding.py（模块 嵌入模型；类别 test；类型 test-coverage；符号 _truncate_prompts, assert_close_prefill_logits, test_prefill_logits, test_matryoshka_embedding）：从端到端 server 测试重构为 HF/SRT parity 测试，并新增 matryoshka embedding 与长输入截断逻辑，测试范式升级的代表。
- test/registered/xpu/test_xpu_reward.py（模块 奖励模型；类别 test；类型 test-coverage；符号 TestXPUReward, assert_close_reward_scores, test_reward_scores）：新增 reward 模型对比测试，验证 Skywork-Reward-V2-Qwen3-0.6B 的分数一致性，是 prefill-only 覆盖的一环。

关键符号：format_prompt, yes_no_token_ids, score_from_token_logprobs, _extract_scores, _assert_close_scores, _hf_probs, _srt_probs, assert_close_reward_scores, _truncate_prompts, assert_close_prefill_logits, test_matryoshka_embedding

关键源码片段

test/registered/xpu/test_xpu_classification.py

新增分类任务测试；review 讨论集中在显存双占与概率容差设计上，最终改为顺序执行 HFRunner/SRTRunner，是测试设计调整的直接体现。

```
class TestXPUClassification(CustomTestCase):
    def _hf_probs(self):
        """HuggingFace 序列分类模型的参考概率。"""
        with HFRunner(
            MODEL_PATH, torch_dtype=TORCH_DTYPE, model_type="cross_encoder"
```

```
    ) as hf_runner:
        hf_scores = hf_runner.forward(PROMPTS).scores

    # 对原始 logits 取 softmax，概率对 bf16 舍入不敏感，
    # 远优于直接比较原始 logits 的稳定性。
    probs = []
    for row in hf_scores:
        tensor = row if torch.is_tensor(row) else torch.tensor(row)
        tensor = tensor.float().flatten()
        probs.append(torch.softmax(tensor, dim=-1))
    return probs

def _srt_probs(self):
    """SRT 侧通过 embedding-mode encode 接口取分类 logits。

    SRT 的 classify 路径复用 embedding 模式的输出（class-logit 向量），
    因此这里用 model_type="embedding" 并读取 embed_logits。
    """
    with SRTRunner(
        MODEL_PATH,
        tp_size=TP_SIZE,
        torch_dtype=TORCH_DTYPE,
        model_type="embedding",
        attention_backend="intel_xpu",
        trust_remote_code=True,
    ) as srt_runner:
        srt_logits = srt_runner.forward(PROMPTS).embed_logits

    probs = []
    for row in srt_logits:
        tensor = row if torch.is_tensor(row) else torch.tensor(row)
        tensor = tensor.float().flatten()
        probs.append(torch.softmax(tensor, dim=-1))
    return probs

def test_classification_logits(self):
    hf_probs = self._hf_probs()
    srt_probs = self._srt_probs()
    self.assertEqual(len(hf_probs), len(PROMPTS))
    self.assertEqual(len(srt_probs), len(PROMPTS))

    for index, (hf_row, srt_row) in enumerate(zip(hf_probs, srt_probs)):
        # 形状一致 + 概率整体接近 + top class 一致，三重校验。
        self.assertEqual(
            srt_row.shape, hf_row.shape, f"probability shape mismatch at sample {index}"
        )
        max_abs_diff = torch.max(torch.abs(hf_row - srt_row)).item()
        self.assertLess(
            max_abs_diff,
```

```
            PROB_TOLERANCE,
            f"classification probs diverged at sample {index}: {max_abs_diff}",
        )
        hf_pred = int(torch.argmax(hf_row).item())
        srt_pred = int(torch.argmax(srt_row).item())
        self.assertEqual(hf_pred, srt_pred, f"top class mismatch at sample {index}")
```

评论区精华

核心讨论集中在两处：

- 显存双占问题：mingfeima 指出 classification 测试中 SRT server 常驻期间 HFRunner 又在同一块 XPU 上加载 1.5B 模型，建议像 embedding/reward 测试那样改为顺序执行以避免双模型占用显存。该问题已在最终代码中解决，HFRunner 的 with 块结束后才启动 SRTRunner。
- cross-encoder 的 backend/dtype 选择：arathi-hlab 质疑 BAAI/bge-reranker-v2-m3 为何不用 intel_xpu backend 且使用 fp32。gaopengff 回复：模型约 2.3GB，B580 显存足够；但 cross-encoder 输出原始 logit，对累加误差非常敏感，融合内核（intel_xpu、flashinfer）的归约顺序与 eager 参考不同，bf16 下会超标 1e-2 容差，因此沿用 CUDA ut 的 triton + fp32 配置与参考口径保持一致。
  - classification 测试显存双占问题 (performance): 已解决：最终代码中 HFRunner 的 with 块结束后才启动 SRTRunner，二者顺序执行。
  - cross-encoder rerank 为何使用 triton + fp32 而非 intel_xpu + bf16 (correctness): 维持 triton + fp32，与 CUDA 参考测试（test/registered/prefill_only）保持口径一致。

风险与影响

- 风险：风险整体较低，本 PR 仅包含测试文件变更，未触及任何生产代码路径。需关注的几点：
- 外部依赖风险：5 个测试均需从 HuggingFace Hub 下载模型与 tokenizer，模型可用性或网络波动会直接导致 CI 失败；test_xpu_classification.py 还启用了 trust_remote_code，依赖社区模型的自定义实现。
- 数值容差偏宽：reward 测试的 TOLERANCE 为 1.5e-1，远大于 rerank 的 1e-2 与 embedding 的 1e-3，可能掩盖部分精度回归。
- 覆盖盲区：cross-encoder rerank 明确绕开 intel_xpu backend，若未来该 backend 支持 fp32，现有测试不会自动覆盖，需要人工跟进补充。
- 长文本断言跳过：embedding 测试对超过 1000 字符的 prompt 跳过相似度断言，长文本精度回归存在漏检可能。
- 影响：对最终用户与生产服务无直接影响（零生产代码变更）。对 Intel XPU 平台的验证能力提升明显：stage-b-test-1-gpu-xpu suite 从单一 embedding 端到端用例扩展为覆盖 embedding、classification、reward、decoder-only rerank、cross-encoder rerank 五类 prefill-only 场景的回归基线。对团队而言，"HF 参考 vs SRT 输出 " 的对比测试范式（HFRunner/SRTRunner 顺序执行、概率 / 分数容差设计）可复用到其他硬件平台的 prefill-only 模型验证。

- 风险标记：纯测试变更无生产风险，依赖外部模型下载，reward 容差偏宽，cross-encoder 避开 intel_xpu backend，长文本断言跳过

关联脉络

- PR #29143 Add intel_xpu to DETERMINISTIC_ATTENTION_BACKEND_CHOICES: 同为 Intel XPU attention backend 相关演进，本 PR 的 prefill-only 测试大量依赖 intel_xpu attention backend，是 XPU 后端能力验证的延续。