PR #28908 完整报告

sgl-project/sglang

[Intel XPU] Initially add nightly GSM8K accuracy tests for Llama-3.1-8B (TP=2) and Qwen3-32B (TP=4)

合并时间：2026-07-01 16:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28908

执行摘要

- 一句话：为 Intel XPU 添加 nightly GSM8K 准确性测试
- 推荐动作：值得关注此 PR 的设计模式：通过 Mixin 复用共享评估逻辑、Step Summary 统一输出格式。对于跨后端的测试基础设施团队有借鉴意义。同时注意 continue-on-error 在 nightly 流中的权衡，建议定期审查测试结果。

功能与动机

PR 描述指出："Wire up the Intel XPU nightly flow on intel-bmg-nightly. Every suite/job is registered via register_xpu_ci(... nightly=True) and dispatched by test/run_suite.py." 目的是确保 Intel XPU 后端上的模型准确性持续可验证，与 AMD/NVIDIA 夜间测试保持一致。

实现拆解

1. 创建共享测试基类：新增 python/sglang/test/xpu/simple_eval_gsm8k_xpu_mixin.py，定义 SimpleEvalGSM8KXPUMixin，封装 XPU 服务器启动（setUpClass）、清理（tearDownClass）和 GSM8K 评估（test_gsm8k），并通过可覆盖的类属性（model, tp_size, num_examples, num_threads, max_tokens 等）实现灵活配置。
2. 提供 GitHub Step Summary 工具：新增 python/sglang/test/xpu/test_xpu_utils.py，包含 write_results_to_github_step_summary 函数，将测试结果渲染为 Markdown 表格并写入 $GITHUB_STEP_SUMMARY，格式与 AMD/Ascend 夜间测试一致。
3. 编写具体测试类：在 test/registered/xpu/llm_models/ 下新增 test_xpu_llama_3_1_8b.py 和 test_xpu_qwen3_32b.py，继承 SimpleEvalGSM8KXPUMixin 并调用 register_xpu_ci 分别注册到 nightly-xpu-2-gpu 和 nightly-xpu-4-gpu 套件，设置相应的准确率阈值（0.80 和 0.85）和启动参数。
4. 修改 GitHub Actions 工作流：重写 nightly-test-intel.yml，添加 nightly-xpu-2-gpu 和 nightly-xpu-4-gpu 作业，运行在 intel-bmg-nightly 标签的 Runner 上，并新增 continue_on_error 输入（默认 true），使定时触发的夜间测试在部分失败时继续。
5. 注册夜间套件：在 test/run_suite.py 的 NIGHTLY_SUITES 映射中为 HWBackend.XPU 添加 nightly-xpu-2-gpu 和 nightly-xpu-4-gpu，确保 run_suite 能发现并调度这些测试。

关键文件：

- python/sglang/test/xpu/simple_eval_gsm8k_xpu_mixin.py（模块 测试 Mixin；类别 test；类型 test-coverage；符号 SimpleEvalGSM8KXPUMixin, setUpClass, tearDownClass, test_gsm8k）：核心共享基类，定义 XPU GSM8K 测试的完整生命周期（启动、评估、清理）。
- python/sglang/test/xpu/test_xpu_utils.py（模块 工具函数；类别 test；类型 test-coverage；符号 _write_header_once, write_results_to_github_step_summary, fmt）：提供 GitHub Step Summary 写入工具，使 XPU 测试输出与 AMD/Ascend 夜间测试一致的 Markdown 表格。
- test/registered/xpu/llm_models/test_xpu_qwen3_32b.py（模块 Qwen3 测试；类别 test；类型 test-coverage；符号 TestQwen3_32BXPU）：Qwen3-32B GSM8K 夜间测试，TP=4，注册到 nightly-xpu-4-gpu 套件。
- test/registered/xpu/llm_models/test_xpu_llama_3_1_8b.py（模块 Llama 测试；类别 test；类型 test-coverage；符号 TestLlama31_8BInstructXPU）：Llama-3.1-8B-Instruct GSM8K 夜间测试，TP=2，注册到 nightly-xpu-2-gpu 套件。
- .github/workflows/nightly-test-intel.yml（模块 工作流；类别 infra；类型 infrastructure）：定义了两个 XPU nightly job（2-GPU 和 4-GPU），支持 continue-on-error，调度在 intel-bmg-nightly runner。
- test/run_suite.py（模块 套件注册；类别 test；类型 test-coverage）：注册新的 XPU nightly 套件到 HWBackend.XPU 映射。

关键符号：SimpleEvalGSM8KXPUMixin.setUpClass, SimpleEvalGSM8KXPUMixin.test_gsm8k, write_results_to_github_step_summary, TestLlama31_8BInstructXPU, TestQwen3_32BXPU

关键源码片段

python/sglang/test/xpu/simple_eval_gsm8k_xpu_mixin.py

核心共享基类，定义 XPU GSM8K 测试的完整生命周期（启动、评估、清理）。

```
"""SimpleEvalGSM8KXPUMixin: XPU 后端 GSM8K 评估共享基类。

子类只需设置 model、tp_size、accuracy 等类属性即可运行。
"""

import os
import subprocess
from abc import ABC
from types import SimpleNamespace

from sglang.srt.utils import kill_process_tree
from sglang.test.run_eval import run_eval
from sglang.test.test_utils import (
    DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
    DEFAULT_URL_FOR_TEST,
    popen_launch_server,
    write_github_step_summary,
```

```
)
from sglang.test.xpu.test_xpu_utils import write_results_to_github_step_summary


class SimpleEvalGSM8KXPUMixin(ABC):
    # 子类可覆盖的配置：
    model: str = ""
    tp_size: int = 1
    num_examples: int | None = 200 # GSM8K 样本数（默认 200，可设为 None 跑全量）
    num_threads: int = 1 # 并发线程数（TP>=2 时 Level Zero 驱动可能出现问题，因此默认单线程）
    max_tokens: int = 512 # 最大生成长度（缩短以降低 prefill-decode 切换频率）
    timeout_for_server_launch: int = DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH

    other_args: list[str] = [
        "--device", "xpu",
        "--attention-backend", "intel_xpu",
        "--dtype", "bfloat16",
        "--trust-remote-code",
        "--disable-overlap-schedule",
        "--disable-radix-cache",
    ]

    @classmethod
    def setUpClass(cls):
        cls.base_url = DEFAULT_URL_FOR_TEST
        env = {**os.environ, **(cls.env or {})}
        args = list(cls.other_args) + ["--tp-size", str(cls.tp_size)]
        try:
            cls.process = popen_launch_server(
                cls.model,
                cls.base_url,
                timeout=cls.timeout_for_server_launch,
                other_args=args,
                env=env,
            )
            cls.server_cmd = subprocess.list2cmdline(cls.process.args)
        except Exception as e:
            write_github_step_summary(f"Failed to launch server for {cls.model}: {e}")
            raise AssertionError(f"Test failed for {cls.model}: {e}")

    @classmethod
    def tearDownClass(cls):
        kill_process_tree(cls.process.pid)

    def test_gsm8k(self):
        accuracy_threshold = getattr(self, "accuracy", 0.0)
        # ... 后续为评估逻辑，包含 run_eval 调用和断言
```

test/registered/xpu/llm_models/test_xpu_qwen3_32b.py

Qwen3-32B GSM8K 夜间测试，TP=4，注册到 nightly-xpu-4-gpu 套件。

```
"""Qwen3-32B GSM8K accuracy on Intel XPU (TP=4)."""
import unittest
import torch
from sglang.test.ci.ci_register import register_xpu_ci
from sglang.test.test_utils import CustomTestCase
from sglang.test.xpu.simple_eval_gsm8k_xpu_mixin import SimpleEvalGSM8KXPUMixin

# 注册到 nightly-xpu-4-gpu 套件，预期耗时 1800 秒
register_xpu_ci(est_time=1800, suite="nightly-xpu-4-gpu", nightly=True)


@unittest.skipUnless(
    torch.xpu.is_available(),
    "Intel XPU not available (torch.xpu.is_available() returned False)",
)
class TestQwen3_32BXPU(SimpleEvalGSM8KXPUMixin, CustomTestCase):
    model = "Qwen/Qwen3-32B"
    tp_size = 4
    accuracy = 0.85 # GSM8K 准确率阈值
    timeout_for_server_launch = 3600 # 64GB BF16 权重加载耗时约 9 分钟

    # 覆盖默认参数：增大 token 上限和显存比例
    other_args = SimpleEvalGSM8KXPUMixin.other_args + [
        "--max-total-tokens", "65536",
        "--mem-fraction-static", "0.8",
    ]


if __name__ == "__main__":
    unittest.main()
```

评论区精华

> continue-on-error 设计意图：mingfeima 质疑该标志会使 workflow 不因测试失败变红，arathi 解释这是 nightly flow 需要运行全部测试，即使部分失败也不应阻塞。
>
> 测试资源开销：mingfeima 担心 Qwen3-32B 模型过大，CI 机器能否承载。arathi 确认 nightly 流程每个模型有独立超时设置（est_time），不影响其他任务。

- continue-on-error 设计意图 (design): 接受 continue-on-error，因为 nightly 需要运行全部测试，失败不应阻塞其他套件。
- Qwen3-32B 测试资源开销 (performance): 在 nightly 范围可接受，通过 est_time 控制。

风险与影响

- 风险：
  - 测试稳定性：GSM8K 评估受随机性影响，阈值设置可能过于宽松，掩盖轻微回归。

- continue-on-error 掩盖失败：默认忽略失败可能导致长期失效测试不被注意，建议配合额外通知机制（如 GitHub Issue 预警）。
- 硬件资源竞争：单台 intel-bmg-nightly 执行多个 Job，若一个 Job 出现问题可能阻塞后续，但因设置了 continue-on-error 仍会继续。
- 超时风险：Qwen3-32B 启动耗时较长（约 9 分钟），虽设置了超时 3600 秒，但未来模型更大时需注意。
- 影响：本次变更仅影响 Intel XPU 后端的夜间测试流水线。对现有 CUDA/AMD 测试无影响，对用户请求路径无任何更改。团队将获得 XPU 后端的回归检测能力，但需关注测试的持续有效性。
- 风险标记：continue-on-error 可能掩盖失败，GSM8K 评估稳定性，测试超时风险，Level Zero 驱动竞争条件

关联脉络

- 暂无明显关联 PR