PR #31751 完整报告

sgl-project/sglang

[XPU] upgrade sglang xpu backend to PyTorch 2.13

合并时间：2026-08-17 18:29

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31751

执行摘要

- 一句话：XPU 后端升级至 PyTorch 2.13 与 oneAPI 2026
- 推荐动作：值得精读，重点看两点：(1) 平台依赖升级如何通过 Dockerfile 与 pyproject 双契约锁版本，并用 CI 实测结果对上游回归做定性而非疲于修 bug；(2) assert_equal 的设计是 " 平台语义差异适配 " 而非 " 掩盖错误 "——它把并列分数下的索引不确定性显式建模为可接受偏差。建议 XPU 负责人将驱动版本约束写入发布文档，并跟进 intel/torch-xpu-ops#4396 的修复状态。

功能与动机

PR body 明确说明 "update pytorch to 2.13 for xpu and update other relation. The oneAPI version should be 2026.0 after updated."，即跟随 PyTorch XPU wheel 与 Intel oneAPI 2026.0 的节奏升级后端运行时。讨论中 arathi-hlab 实测暴露了 torch.topk 在 XPU 上的 SIGSEGV，YangKai0616 也指出 torch.nonzero / torch.unique 在特定驱动下的截断问题，驱动版本必须同步更新才能规避，这构成了升级的附加动机。

实现拆解

1. 依赖契约升级（python/pyproject_xpu.toml）：将 torch==2.12.0+xpu 提升为 torch==2.13.0+xpu，配套更新 torchvision==0.27.0+xpu -> 0.28.0+xpu、torchcodec==0.12.0 -> 0.13.0，并同步修改版本注释。原因是 PyTorch XPU wheel 与 torchvision / torchcodec 存在强版本配对关系，三件套必须一起对齐，否则 pip 解析会引入 CUDA 版本或 ABI 冲突。该文件是 XPU pip 安装的依赖契约源头，Docker 构建时会被复制为 pyproject.toml 使用。
2. 镜像与驱动基建（docker/xpu.Dockerfile）：基础镜像从 intel/deep-learning-essentials:2025.3.2-0-devel-ubuntu24.04 切到 2026.0.0-devel-ubuntu24.04；Level-Zero UMD（COMPUTE_RUNTIME_VERSION）、IGC 编译器、GMM 三件运行时从 26.05 / 2.28 / 22.9 升级到 26.18.38308.1 / 2.34.4 / 22.10.0，并保留注释说明这些版本需与宿主机 xe KMD 保持 lockstep。安装命令重构为先装 PyTorch 2.13 XPU 三件套，再 pip install ".[dev,diffusion]"，使镜像一次到位包含 diffusion 依赖，替代原先需要额外安装 diffusion 的流程。
3. topk 单测适配（test/registered/xpu/test_topk.py）：新增 torch.use_deterministic_algorithms(True) 以增强可复现性；新增 assert_equal 辅助函数，用 " 索引集合差集 + 对应 score 值比对 " 替代 torch.testing.assert_close 的逐元素权

重比对。原因是 PyTorch 2.13 的 XPU torch.topk 对并列分数返回的索引顺序与参考实现可能不同，逐元素对比会误报；改为集合容差后即使并列索引互换也能通过，并保留 max_permit_error 控制允许误差。

4. 文档与安装方式（docs/docs/sglang-diffusion/installation.mdx、docs/docs/hardware-platforms/xpu.mdx）：diffusion 安装文档删除了在 XPU 上直接 pip install -e "python[diffusion]" 的段落，改为说明官方 Docker 镜像已内置 diffusion 依赖；XPU 安装文档同步更新 PyTorch 三件套版本号。这与 Dockerfile 中 ".[dev,diffusion]" 的改动形成闭环。
5. 验证与回退（CI 实测）：基于 run_suite.py 的 stage-a（1 GPU）与 stage-b（1 GPU）套件验证，官方实测 stage-a 4/4 通过、stage-b 9/11 通过；两个失败分别在 test_topk（torch.topk 在 XPU 上 SIGSEGV）与另一个与动态 shape 相关的用例，均复现为 torch 2.13 上游回归而非 PR 引入。期间曾临时 disable 掉 test_deepseek_ocr.py 并引发 review 质询，最终通过更新 XPU driver 修复并 revert 该禁用。

关键文件：

- docker/xpu.Dockerfile（模块 镜像构建；类别 infra；类型 infrastructure）：升级落地的主要载体：基础镜像切到 oneAPI 2026.0.0，Level-Zero UMD / IGC / GMM 驱动三件套整体升级，安装命令改为同时装入 dev 与 diffusion 依赖，使镜像成为 XPU 部署的一致基线。
- python/pyproject_xpu.toml（模块 依赖配置；类别 config；类型 configuration）：XPU pip 依赖契约核心：torch / torchvision / torchcodec 三件套版本严格配套，是整个升级的版本矩阵依据，Docker 构建时直接复用此文件。
- test/registered/xpu/test_topk.py（模块 单元测试；类别 test；类型 test-coverage；符号 assert_equal）：针对 PyTorch 2.13 XPU topk 并列索引顺序不确定的适配：新增 assert_equal 集合容差断言替代 torch.testing.assert_close，并开启确定性算法，是本 PR 最有技术含量的配套改动。
- docs/docs/sglang-diffusion/installation.mdx（模块 文档；类别 docs；类型 documentation）：diffusion 安装方式从独立的 pip 命令改为官方 Docker 路径，与 Dockerfile 内 .[dev,diffusion] 改动形成闭环，影响 XPU 多模态用户的部署文档。
- docs/docs/hardware-platforms/xpu.mdx（模块 文档；类别 docs；类型 documentation）：同步更新 XPU 基准安装文档中的 PyTorch 三件套版本号，保证文档与 pyproject / Dockerfile 一致。
- test/registered/xpu/test_deepseek_ocr.py（模块 单元测试；类别 test；类型 test-coverage）：review 焦点文件：升级期间曾临时 disable 该用例，arathi-hlab 要求说明理由、mingfeima 追问纯 torch 复现，最终由升级 XPU driver 修复并 revert 禁用；反映升级对 OCR 路径的驱动敏感性。

关键符号：assert_equal

关键源码片段

docker/xpu.Dockerfile

升级落地的主要载体：基础镜像切到 oneAPI 2026.0.0，Level-Zero UMD / IGC / GMM 驱动三件套整体升级，安装命令改为同时装入 dev 与 diffusion 依赖，使镜像成为 XPU 部署的一致基线。

```
# docker/xpu.Dockerfile
# 基础镜像切换到 oneAPI 2026.0.0，与 PyTorch 2.13 XPU wheel 配套。
FROM intel/deep-learning-essentials:2026.0.0-devel-ubuntu24.04

# Level-Zero UMD + IGC/GMM 版本锁定：
# 滚动 PPA 曾因版本漂移在 B580 上引入 libze 问题（sgl-kernel-xpu#296），
# 三件套需与宿主机 xe KMD 保持 lockstep，必要时用 --build-arg 覆盖。
ARG COMPUTE_RUNTIME_VERSION=26.18.38308.1
ARG IGC_VERSION=2.34.4+21428
ARG GMM_VERSION=22.10.0

# 先安装与 pyproject_xpu.toml 一致的 PyTorch XPU 三件套，
# 再安装 sglang 本体；".[dev,diffusion]" 让镜像自带 diffusion 依赖。
RUN pip install --no-cache-dir torch==2.13.0+xpu torchvision==0.28.0+xpu \
        torchaudio==2.11.0+xpu --index-url https://download.pytorch.org/whl/xpu && \
    pip install --no-cache-dir msgspec blake3 py-cpuinfo compressed_tensors \
        gguf partial_json_parser einops tabulate --root-user-action=ignore && \
    cd sglang/python && \
    cp pyproject_xpu.toml pyproject.toml && \
    pip install --no-cache-dir ".[dev,diffusion]" \
        --extra-index-url https://download.pytorch.org/whl/xpu
```

python/pyproject_xpu.toml

XPU pip 依赖契约核心：torch / torchvision / torchcodec 三件套版本严格配套，是整个升级的版本矩阵依据，Docker 构建时直接复用此文件。

```
# python/pyproject_xpu.toml
# PyTorch XPU 版本三件套必须严格配套：torch 2.13.0 对应
# torchvision 0.28.0 与 torchcodec 0.13.0，任何一项错配都会
# 引入 CUDA 版本或 ABI 冲突。
"torch==2.13.0+xpu",
"torchaudio==2.11.0+xpu",
"torchcodec==0.13.0 ; sys_platform != 'linux' or (sys_platform == 'linux' \
    and platform_machine != 'aarch64' and platform_machine != 'arm64' \
    and platform_machine != 'armv7l')",
"torchvision==0.28.0+xpu",
```

test/registered/xpu/test_topk.py

针对 PyTorch 2.13 XPU topk 并列索引顺序不确定的适配：新增 assert_equal 集合容差断言替代 torch.testing.assert_close，并开启确定性算法，是本 PR 最有技术含量的配套改动。

```
# test/registered/xpu/test_topk.py
# PyTorch 2.13 起 XPU torch.topk 对并列分数返回的索引顺序与参考实现不一致，
# 因此关闭逐元素断言，改用集合容差的方式比较 topk 结果。
from typing import Optional
```

```
import torch

# 开启确定性算法，让同一 seed 下的结果可复现，便于 CI 定位平台差异。
torch.use_deterministic_algorithms(True)


def assert_equal(
    score: torch.Tensor,
    indices_ref: torch.Tensor,
    indices_our: torch.Tensor,
    bs: int,
    k: int,
    seq_len: int,
    topk_indices_offset: Optional[torch.Tensor] = None,
    max_permit_error: int = 0,
):
    """允许并列分数索引互换的 topk 一致性校验。

    逐元素比较在 XPU 上会因并列索引顺序不确定而误报；
    这里改为比较每行索引集合的差集：
    * 若差集非空，再核对被替换索引对应的 score 值是否完全相同；
    * 只有 score 不一致才累计为 wrong_values 并与 max_permit_error 比较。
    """
    indices_our_cpu = indices_our.cpu().tolist()
    indices_ref_cpu = indices_ref.cpu().tolist()

    wrong_values = 0
    for i in range(bs):
        more = set(indices_our_cpu[i]) - set(indices_ref_cpu[i])
        less = set(indices_ref_cpu[i]) - set(indices_our_cpu[i])
        offset = (
            topk_indices_offset[i].item() if topk_indices_offset is not None else 0
        )
        if more or less:
            # 并列分数下 topk 选哪个索引都等价，因此比较差集对应分数值。
            more_values = sorted(score[i, idx - offset].item() for idx in more)
            less_values = sorted(score[i, idx - offset].item() for idx in less)
            if more_values != less_values:
                wrong_values += len(more)
                print(
                    f"{bs=}, {k=}, {seq_len=}, {i=}, {more=}, {less=} failed"
                    f", with {more_values=}, {less_values=}"
                )
    assert wrong_values <= max_permit_error, f"{wrong_values=}, {max_permit_error=}"


# 在 test_grouped_topk 中，原先的 torch.testing.assert_close 会同时比较
# topk_weights 的散布密集张量；XPU 上并列索引顺序不同导致误报，改为：
assert_equal(
```

```
    gating_output,
    ref_topk_ids[:, :topk_routed],
    topk_ids[:, :topk_routed],
    bs=len(bs),
    k=topk_value,
    seq_len=seq_len,
)
```

评论区精华

> polisettyvarma：we already have PR https://github.com/sgl-project/sglang/pull/31031 for pytorch upgrade on XPU @yuchengliu1 let's use older PR —— 提示本 PR 与 #31031 工作线重叠，最终本分支（含驱动与 diffusion 配套）胜出合入。
>
> arathi-hlab（实测验证）：stage-a 4/4 通过，stage-b 9/11 通过；test_biased_grouped_topk 的失败可缩减为 3 行纯 PyTorch：x.topk(2, dim=-1) 在 XPU 上直接 SIGSEGV，CPU 上正常 —— 用最小复现把问题定性为上游回归，而非 sglang 代码缺陷。
>
> YangKai0616：指定 XPU 驱动下 torch.nonzero 和 torch.unique 可能返回截断或空结果，导致 boolean indexing 等动态 shape 操作结果错误，workaround 见 intel/torch-xpu-ops#4396。
>
> mingfeima：are we able to reproduce this issue via pure torch? i mean without sglang —— 推动问题定位到尽可能小的复现面。
>
> yuchengliu1：This failure fixed by update XPU driver；torch.nonzero、torch.unique 的报错也能通过更新 XPU driver 解决。

- 已有 PR #31031 做同样的 XPU PyTorch 升级，建议复用旧 PR (question): 本 PR 仍然推进并最终合入，说明新分支（含驱动升级与 diffusion 配套）被采纳；旧 PR #31031 的后续状态在本次材料中未提供。
- stage-a/stage-b 实测与 torch.topk 在 XPU 上的 SIGSEGV (testing): 失败被定性为 torch 2.13.0+xpu 上游回归，非 PR 引入；测试环境继续保留该版本作为已知限制。
- deepseek_ocr UT 禁用理由与纯 torch 复现 (question): yuchengliu1 答复该失败由更新 XPU driver 修复，后续 revert 了禁用提交。
- torch.nonzero / torch.unique 在 2.13 XPU 驱动下返回截断结果 (correctness): yuchengliu1 表示可通过更新 XPU driver 解决；PR 内未显式变更驱动规避，存在遗留风险。

风险与影响

- 风险：
  1. MoE 路由路径崩溃隐患：torch.topk 是 sglang/srt/layers/moe/topk.py 路由逻辑的核心算子，arathi-hlab 已用 3 行纯 PyTorch 复现其 XPU SIGSEGV。若用户环境驱动版本与 PR 锁定的 26.18.38308.1 不一致，MoE 模型（如 Nemotron-3、DeepSeek 系列）在运行时有崩溃风险。

1. 动态 shape 静默错误：YangKai0616 指出指定驱动下 torch.nonzero / torch.unique 可能返回截断或空结果，影响 boolean indexing 等动态 shape 操作；PR 内未将这一约束显式写进文档或代码防护，只能依赖驱动更新规避，存在静默算错的窗口。
2. 测试断言能力回退：test_topk.py 的 assert_equal 只校验索引集合与对应 score 值，不再校验 topk_weights 的具体散布值（原 assert_close 同时比较权重与索引），对权重精度回归的感知变弱；虽然 max_permit_error=0 默认不放宽，但语义上从 " 强一致 " 退化为 " 集合一致 "。
3. 强版本绑定与升级门槛：torch==2.13.0+xpu 被精确 pin，用户无法用 pip 平滑升级；Dockerfile 锁定的驱动 / IGC / GMM 三者必须与宿主机 KMD 匹配，任何一项错配都会导致实例不可用或运行期 crash。 - 影响：
   - 用户侧：所有 XPU 部署必须跟随 oneAPI 2026.0 与新版驱动；diffusion 用户从 " 手动 pip 安装 " 改为直接使用官方 Docker 镜像，安装路径收敛。
   - 系统侧：无核心推理算法改动，影响集中在运行时版本与平台适配层；XPU 特有 test_topk.py、test_deepseek_ocr.py 的 CI 行为发生变化。
   - 团队侧：确立了一套 "pyproject 契约 + Docker 镜像 + 容错单测 + 上游回归定性 " 的平台升级范式，对后续 XPU 版本跟进有直接参考价值。
   - 风险标记：XPU 全局版本强绑定，MoE 路由 topk 崩溃风险，上游回归未修复，断言强度回退

关联脉络

- PR #31031 pytorch upgrade on XPU: polisettyvarma 在评论中明确指出已存在 #31031 做 XPU PyTorch 升级，本 PR 是并行分支；最终本 PR（含驱动与 diffusion 配套）合入，体现升级工作线的整合。
- PR #34818 [CI] Install sgl-eval in xeon (CPU) Docker image: 同属 Intel 平台 Docker / CI 基建演进，展示 Intel 平台镜像与 CI 依赖持续迭代，与本 PR 的镜像改造同一条工程线。
- PR #35105 Revert "[AMD] [GLM5] Fuse shared-expert append into aiter grouped-topk (skip per-layer append kernel)": 同属 topk 跨平台稳定性主题；本 PR 的 test_topk 容错断言正是应对不同平台 topk 语义 / 索引顺序差异的通用手段。