PR #13599 完整报告

sgl-project/sglang

Replace hardcoded CUDA device with get_device() for XPU support

合并时间：2026-05-01 07:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/13599

执行摘要

- 一句话：替换硬编码 CUDA 设备为 get_device() 以支持 XPU
- 推荐动作：建议精读。虽然变更本身简单，但它展示了在大型项目中如何逐步引入设备抽象模式。重点关注 llama.py 中的条件分支和 expert_distribution.py 中通过 get_device() 抽象张量设备的方式，这对于未来支持更多硬件（如 NPU、AMD GPU）有参考价值。

功能与动机

PR body 指出：'Replace hardcoded CUDA device with get_device() for XPU support'。Gemini Code Assist 评论总结：'This pull request modernizes the test infrastructure by abstracting device-specific calls. By replacing direct references to "cuda" with a utility function, the tests can now seamlessly adapt to different hardware platforms, such as XPU'。

实现拆解

实现分为四个主要步骤：

1. 模型缓存清理设备抽象（python/sglang/srt/models/llama.py）：在 set_embed_and_head 和 set_embed 方法中，将直接调用 torch.cuda.empty_cache() 和 torch.cuda.synchronize() 替换为条件分支：如果 _is_xpu 为真，则调用 torch.xpu 的对应方法，否则回退到 torch.cuda。同时新增 is_cuda()、is_xpu() 帮助函数导入。
2. 专家分布记录器设备泛化（python/sglang/srt/eplb/expert_distribution.py）：在 _LayerBasedGpuSinglePassGatherer.__init__ 中，将张量 device="cuda" 替换为 device=get_device()，并导入 get_device 函数。这使专家分布记录能够运行在非 CUDA 设备上。
3. FP8 内核测试设备无关化（test/registered/quant/test_fp8_kernel.py）：将测试中所有 torch.rand(..., device="cuda") 替换为 device 变量（通过 get_device() 获取），并在设备能力检查中增加 _is_cuda 判断，对 XPU 跳过计算能力检查。
4. VLM 输入格式测试设备动态选择（test/registered/vlm/test_vlm_input_format.py）：在 setUpClass 中，不再使用 torch.cuda.is_available() 决定设备，而是通过 is_cuda() / is_xpu() 函数判断并设置 torch.device("cuda") 或 torch.device("xpu")，否则回退到 CPU。

关键文件：

- python/sglang/srt/models/llama.py（模块 模型层；类别 source；类型 core-logic；符号 set_embed_and_head, set_embed）：核心推理模型，修改了缓存清理的设备抽象，是本次变更中影响最大的源码文件。
- python/sglang/srt/eplb/expert_distribution.py（模块 专家分配；类别 source；类型 dependency-wiring；符号 _LayerBasedGpuSinglePassGatherer.init）：专家分布记录器使用 get_device() 替代硬编码 'cuda'，属于配套的依赖注入改造。
- test/registered/quant/test_fp8_kernel.py（模块 FP8 内核；类别 test；类型 test-coverage；符号 TestFP8Base, _make_A, _make_B, test_per_token_group_quant_fp8）：FP8 内核测试设备无关化，验证量化 kernel 在非 CUDA 设备上的正确性。
- test/registered/vlm/test_vlm_input_format.py（模块 VLM 测试；类别 test；类型 test-coverage；符号 VLMInputTestBase.setUpClass）：VLM 输入测试动态选择设备，适配 XPU 环境。

关键符号：set_embed_and_head, set_embed, _LayerBasedGpuSinglePassGatherer.init, test_per_token_group_quant_fp8, test_w8a8_block_fp8_matmul, VLMInputTestBase.setUpClass

关键源码片段

python/sglang/srt/models/llama.py

核心推理模型，修改了缓存清理的设备抽象，是本次变更中影响最大的源码文件。

```
def set_embed_and_head(self, embed, head):
    del self.model.embed_tokens.weight
    del self.lm_head.weight
    self.model.embed_tokens.weight = embed
    self.lm_head.weight = head
    # 根据当前设备类型选择正确的缓存清理函数
    if _is_xpu:
        torch.xpu.empty_cache()
        torch.xpu.synchronize()
    else:
        torch.cuda.empty_cache()
        torch.cuda.synchronize()

def set_embed(self, embed):
    # 注：若 draft hidden size 不等于 target hidden size，EAGLE3 无法共享 embed
    if (
        hasattr(self.config, "target_hidden_size")
        and self.config.target_hidden_size != self.config.hidden_size
    ):
        return
    del self.model.embed_tokens.weight
    self.model.embed_tokens.weight = embed
    # 同样根据设备选择缓存清理
    if _is_xpu:
```

```
        torch.xpu.empty_cache()
        torch.xpu.synchronize()
    else:
        torch.cuda.empty_cache()
        torch.cuda.synchronize()
```

python/sglang/srt/eplb/expert_distribution.py

专家分布记录器使用 get_device() 替代硬编码 'cuda'，属于配套的依赖注入改造。

```
class _LayerBasedGpuSinglePassGatherer(_SinglePassGatherer):
    def __init__(self, *args, enable_global_physical_experts: bool, **kwargs):
        super().__init__(*args, **kwargs)

        # 使用 get_device() 替代固定 "cuda" 字符串，实现设备无关
        device = get_device()

        self._enable_global_physical_experts = enable_global_physical_experts
        self._data = torch.zeros(
            (
                self._expert_location_metadata.num_layers,
                (
                    self._expert_location_metadata.num_physical_experts
                    if enable_global_physical_experts
                    else self._expert_location_metadata.num_local_physical_experts
                ),
            ),
            dtype=torch.int,
            device=device, # 动态设备，支持 CUDA/XPU 等
        )
```

test/registered/quant/test_fp8_kernel.py

FP8 内核测试设备无关化，验证量化 kernel 在非 CUDA 设备上的正确性。

```
_device = get_device() # 全局设备变量，替代硬编码 "cuda"

class TestFP8Base(CustomTestCase):
    @staticmethod
    def _make_A(M, K, group_size, out_dtype):
        # 使用 _device 而非固定 "cuda"
        quant_A = torch.rand(
            M, K // group_size, group_size, dtype=torch.float32, device=_device
        )
        # ... 其余逻辑不变
        scale = torch.rand(M, K // group_size, dtype=torch.float32, device=_device)
        return A, quant_A, scale

class TestPerTokenGroupQuantFP8(TestFP8Base):
    def test_per_token_group_quant_fp8(self):
        # 只有 CUDA 且计算能力 < 9 时才跳过
```

```
        if _is_cuda and torch.cuda.get_device_capability()[0] < 9:
            return
        # ... 实际测试逻辑

class TestW8A8BlockFP8Matmul(TestFP8Base):
    def test_w8a8_block_fp8_matmul(self):
        if _is_cuda and torch.cuda.get_device_capability()[0] < 9:
            return
        elif _is_xpu:
            # XPU 不提供类似 CUDA 的计算能力，直接跳过检查
            pass
        else:
            return
        # ... 实际测试逻辑
```

评论区精华

主要讨论来自 Gemini Code Assist 的自动审查：'The changes are consistent and correctly implemented, making the test runners more device-agnostic and enabling support for devices like XPU.' 合入者 mingfeima 表示 'this one is low risk, rebase again since this is a bit old. check ci result again.' 无未解决的反对意见。

- 低风险评估与合入决策 (other): 经过 rebase 和 CI 验证后，PR 被批准合并。

风险与影响

- 风险：风险较低，但仍需注意：
  - 遗漏的 CUDA 调用：llama.py 中仅修改了 set_embed_and_head 和 set_embed 方法，其他可能存在的 torch.cuda 调用（如 torch.cuda.current_device() 等）未被替换，在 XPU 上可能引发错误。
  - get_device() 行为依赖：在测试中直接调用 get_device() 假设设备已初始化，若在无 GPU 环境运行测试可能返回 cpu，但 FP8 内核测试预期在 GPU 上运行，这可能导致测试跳过或失败。不过当前测试已通过 register_cuda_ci 标记为 CUDA 专用，风险可控。
  - XPU 特定路径覆盖：新增的 torch.xpu 调用仅在 _is_xpu 为真时执行，但 torch.xpu 模块是否在 Intel XPU 环境中始终可用未在代码中验证。
  - 影响：对 用户的影响：XPU 用户现在可以在 Llama 模型和专家分布记录功能中使用正确的设备缓存清理，FP8 和 VLM 测试也可以在 XPU 上运行（需相应硬件和驱动）。对 CUDA 用户无影响，原有行为完全保留。对 开发者的影响：提供了一个设备抽象模式的范例，后续可推广到其他模块。整体影响范围小，但为多硬件支持奠定了基础。
  - 风险标记：核心路径变更，设备抽象覆盖不完整，CI 依赖 XPU 环境

关联脉络

- PR #23557 [Intel GPU] Integrate flash_mla_decode in Intel XPU attention backend: 同一 XPU 支持系列，llama.py 和测试文件的修改为 XPU 后端提供了基础设备抽象。

- PR #22236 [Test] Add XPU device support to unit tests: 本 PR 延续了 XPU 测试支持的工作，进一步在 FP8 和 VLM 测试中替换硬编码设备。