PR #24096 完整报告

sgl-project/sglang

Introduce CudaDeviceMixin and CudaSRTPlatform

合并时间：2026-05-16 01:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24096

执行摘要

- 一句话：引入 CudaDeviceMixin 与 CudaSRTPlatform 平台抽象层
- 推荐动作：值得精读。该 PR 是 SGLang 平台抽象层的关键基础设施，设计模式（Mixin + 自动发现、ROCm 继承 CUDA）对多硬件支持有借鉴意义。关注设备操作接口定义与回退逻辑，为后续 OOT 插件扩展提供参考。

功能与动机

SRT 运行时长期以来直接调用 torch.cuda 接口，导致平台紧密耦合 CUDA，难以扩展至 AMD ROCm 或其他硬件。该 PR 旨在构建平台抽象骨架，使设备操作多态化，并为后续 OOT（Out-of-Tree）插件机制奠定基础。Issue 评论中 alexnails 提到，此 PR 遵循与各硬件团队评审过的设计文档，但与其他提出的设计（如 #20372）略有不同。

实现拆解

1. 定义 CUDA 设备操作类：在 python/sglang/srt/platforms/cuda.py 中新增 CudaDeviceMixin 类，继承 DeviceMixin，实现所有与 CUDA 设备相关的操作方法，如 get_device_total_memory、get_current_memory_usage、get_device、set_device、empty_cache、synchronize 等。同时定义 CudaSRTPlatform，继承 CudaDeviceMixin 和 SRTPlatform，并覆盖 supports_fp8、support_cuda_graph 等能力标志。
2. ROCm 适配类：在 python/sglang/srt/platforms/rocm.py 中新增 RocmDeviceMixin，直接继承 CudaDeviceMixin，仅覆盖 _enum 和 device_name，因为 PyTorch 中 HIP 设备仍然通过 torch.cuda API 暴露。RocmSRTPlatform 继承自 RocmDeviceMixin 和 SRTPlatform，但保留默认的保守能力标志。
3. 增强平台发现机制：修改 python/sglang/srt/platforms/__init__.py，添加 _is_cuda_available() 和 _is_rocm_available() 辅助函数，分别检测纯 CUDA（torch.cuda.is_available() 且 torch.version.hip is None）和 ROCm（torch.cuda.is_available() 且 torch.version.hip is not None）。在 _resolve_platform() 回退路径中，当无插件激活时按顺序检测 CUDA/ROCm，并返回对应的 SRTPlatform 实例。
4. 迁移 torch.cuda 调用：在多个模块中将硬编码的 torch.cuda.empty_cache()、torch.cuda.synchronize() 替换为 current_platform.empty_cache()、current_platform.synchronize()。涉及文件包括 loader.py（模型权重加载与卸载）、memory_pool.py（内存池管理）、scheduler.py（调度器）、model_runner.py（模型运行器）。此迁移在提交历史中曾回退后又重新应用，以确保 AMD 平台不会因缺少

RocmSRTPlatform 而静默无操作。

5. 单元测试覆盖：在 test/registered/unit/platforms/test_platform_interface.py 中新增 TestCudaDeviceMixin 类，使用 mock 验证 CudaSRTPlatform 各方法委托到正确的 torch.cuda 函数。同时添加基础平台身份测试 test_base_device_identity_stays_unspecified，验证 SRTPlatform 基类不声称任何具体设备。

关键文件：

- python/sglang/srt/platforms/cuda.py（模块 平台层；类别 source；类型 core-logic；符号 CudaDeviceMixin, get_device_total_memory, get_current_memory_usage, get_device）：核心新增文件，定义了 CudaDeviceMixin 和 CudaSRTPlatform，是平台抽象层的基础。
- python/sglang/srt/platforms/rocm.py（模块 ROCm 适配；类别 source；类型 core-logic；符号 RocmDeviceMixin, RocmSRTPlatform）：新增 ROCm 适配类，继承 CUDA 设备操作实现，展示了平台抽象层的设计模式。
- test/registered/unit/platforms/test_platform_interface.py（模块 测试；类别 test；类型 test-coverage；符号 test_base_device_identity_stays_unspecified, TestCudaDeviceMixin, test_default_get_device_returns_cuda_device, test_cuda_platform_identity）：新增大量单元测试，覆盖 CudaDeviceMixin 各方法的默认行为，确保抽象层符合预期。
- python/sglang/srt/platforms/__init__.py（模块 平台发现；类别 source；类型 core-logic；符号 _is_cuda_available, _is_rocm_available）：修改平台发现逻辑，添加 CUDA 和 ROCm 检测函数，实现自动回退。
- python/sglang/srt/platforms/device_mixin.py（模块 基类；类别 source；类型 dependency-wiring）：调整导入顺序和模块层次，将 torch 等导入提升至顶层，一致性调整。
- python/sglang/srt/model_loader/loader.py（模块 模型加载；类别 source；类型 data-contract）：将 torch.cuda.empty_cache/synchronize 替换为 current_platform 调用，是迁移的主要目标之一。

关键符号：CudaDeviceMixin.get_device_total_memory, CudaDeviceMixin.get_current_memory_usage, CudaDeviceMixin.get_device, CudaDeviceMixin.set_device, CudaDeviceMixin.get_device_name, CudaDeviceMixin.get_device_uuid, CudaDeviceMixin.get_device_capability, CudaDeviceMixin.empty_cache, CudaDeviceMixin.synchronize, CudaDeviceMixin.get_available_memory, CudaDeviceMixin.seed_everything, RocmDeviceMixin, RocmSRTPlatform, _resolve_platform, _is_cuda_available, _is_rocm_available

关键源码片段

python/sglang/srt/platforms/cuda.py

核心新增文件，定义了 CudaDeviceMixin 和 CudaSRTPlatform，是平台抽象层的基础。

```
class CudaDeviceMixin(DeviceMixin):
```

```
'''CUDA implementation of the shared device operations.'''
_enum = PlatformEnum.CUDA
device_name = 'cuda'
device_type = 'cuda'

def get_device_total_memory(self, device_id=0):
    # 获取指定设备的总显存
    return int(torch.cuda.get_device_properties(device_id).total_memory)

def get_current_memory_usage(self, device=None):
    # 获取当前显存使用量（峰值分配）
    return float(torch.cuda.max_memory_allocated(device))

def get_device(self, local_rank):
    # 根据 local_rank 构造 CUDA 设备对象
    return torch.device('cuda', local_rank)

def set_device(self, device):
    torch.cuda.set_device(device)

def get_device_name(self, device_id=0):
    return str(torch.cuda.get_device_name(device_id))

def get_device_uuid(self, device_id=0):
    # UUID 格式因平台而异，此处保持 CUDA 特定实现
    return str(torch.cuda.get_device_properties(device_id).uuid)

def get_device_capability(self, device_id=0):
    major, minor = torch.cuda.get_device_capability(device_id)
    return DeviceCapability(major, minor)

def empty_cache(self):
    torch.cuda.empty_cache()

def synchronize(self):
    torch.cuda.synchronize()

def get_available_memory(self, device_id=0):
    return torch.cuda.mem_get_info(device_id)

def get_torch_distributed_backend_str(self):
    return 'nccl'

@classmethod
def seed_everything(cls, seed=None):
    if seed is not None:
        super().seed_everything(seed)
        torch.cuda.manual_seed_all(seed)
```

```
class CudaSRTPlatform(CudaDeviceMixin, SRTPlatform):
    '''Default in-tree CUDA SRT platform.'''
    def supports_fp8(self) -> bool:
        return True

    def support_cuda_graph(self) -> bool:
        return True

    def support_piecewise_cuda_graph(self) -> bool:
        return True
```

python/sglang/srt/platforms/rocm.py

新增 ROCm 适配类，继承 CUDA 设备操作实现，展示了平台抽象层的设计模式。

```
'''ROCm device operations for the SRT platform layer.

PyTorch exposes ROCm through the same torch.cuda.* API surface as CUDA
(HIP is a binary shim, and torch.device('rocm') does not exist). So
RocmDeviceMixin inherits all device ops from CudaDeviceMixin and
only overrides identity (_enum, device_name).
'''

from sglang.srt.platforms.cuda import CudaDeviceMixin
from sglang.srt.platforms.device_mixin import PlatformEnum
from sglang.srt.platforms.interface import SRTPlatform


class RocmDeviceMixin(CudaDeviceMixin):
    '''ROCm device ops — identical surface to CUDA via torch.cuda's HIP shim.'''

    _enum: PlatformEnum = PlatformEnum.ROCM
    device_name: str = 'rocm'
    # device_type stays 'cuda' — torch.device('cuda') is the only valid
    # device-type string for HIP devices in PyTorch.


class RocmSRTPlatform(RocmDeviceMixin, SRTPlatform):
    '''Default in-tree ROCm SRT platform.

    Capability flags (supports_fp8, support_cuda_graph, support_piecewise_cuda_graph)
    keep the conservative SRTPlatform defaults rather than mirroring CudaSRTPlatform.
    They are currently only consulted in OOT branches gated on is_out_of_tree(),
    so the defaults are behaviorally inert for the in-tree ROCm path. A follow-up
    that migrates AMD-specific gating off legacy is_hip() should set these here.
    '''
```

评论区精华

在 Review 中，AgainstEntropy 提出了多个关键意见：

- CudaSRTPlatform 位置：建议将 CudaSRTPlatform 从 interface.py 移至 cuda.py，因为 interface.py 应只包含接口。该建议被 alexnails 接受并实施。
- 方法覆盖完整性：指出 get_device_uuid 和 seed_everything 在初始实现中未覆盖。alexnails 回应 get_device_uuid 因其他平台 UUID 格式不同而无法泛化，seed_everything 随后被移入 Mixin 中。
- 能力标志覆盖：要求 CudaSRTPlatform 覆盖 supports_fp8、support_cuda_graph、support_piecewise_cuda_graph。最终实现中这些方法均返回 True。
  - 此外，在提交历史中，作者曾回退 current_platform.empty_cache() 迁移，待 RocmSRTPlatform 就绪后才重新应用，以防止 AMD 路径静默无操作。
  - CudaSRTPlatform 定义位置 (design): alexnails 接受并移动，最终 CudaSRTPlatform 定义在 cuda.py。
  - get_device_uuid 和 seed_everything 覆盖 (correctness): seed_everything 被实现；get_device_uuid 保留在 CUDA 专用类中，不在 DeviceMixin 基类强制。
  - 能力标志覆盖（supports_fp8 等） (design): alexnails 实现，在 cuda.py 中 CudaSRTPlatform 返回 True。

风险与影响

- 风险：平台误判风险：自动检测依赖 torch.cuda.is_available() 和 torch.version.hip，但某些环境可能同时存在 CUDA 和 ROCm 驱动（罕见），或通过模拟设备导致误判。建议保留 SGLANG_PLATFORM 环境变量作为覆盖。

迁移完整性风险：虽然迁移了 empty_cache 和 synchronize，但其他 torch.cuda 函数可能未被替换（例如直接使用 torch.cuda.current_device()），存在残留硬编码。

OOT 平台兼容性：当前 in-tree 的 CudaSRTPlatform 和 RocmSRTPlatform 与 OOT 插件共存逻辑可能引入优先级问题，若 OOT 插件与 in-tree 平台同时激活，len(activated)>1 会报错，要求设置 SGLANG_PLATFORM。

- 影响：用户影响：无功能性变化，向后兼容。用户无需变更配置，但未来可通过 SGLANG_PLATFORM 环境变量选择平台。

系统影响：架构上解耦了设备操作，便于添加新硬件支持。但需确保所有 CUDA 调用点被枚举替换。

团队影响：需要维护设备操作接口，新增平台只需实现 DeviceMixin 子类和 SRTPlatform 子类。

- 风险标记：核心路径变更，平台兼容性，迁移完整性，自动检测假设

关联脉络

- 暂无明显关联 PR