PR #26385 完整报告

sgl-project/sglang

Introduce CpuDeviceMixin and CpuSRTPlatform

合并时间：2026-06-18 08:41

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26385

执行摘要

- 一句话：为 CPU 引擎引入平台抽象层
- 推荐动作：该 PR 是精心设计的预备基础设施，值得精读。关键设计决策（架构分支、内存查询契约、empty_cache 实现权衡、分布式后端选择等）有清晰的文档注释。对于计划为其他设备添加平台类的开发者，参考 CpuDeviceMixin 和 CpuSRTPlatform 的实现模式和测试方法非常有价值。

功能与动机

镜像 PR #24096（添加 CudaDeviceMixin/CudaSRTPlatform 和 RocmDeviceMixin/RocmSRTPlatform）进行 CPU 平台抽象。此前 CPU 主机只能回退到抽象基类 SRTPlatform，其 [Active] 方法抛出 NotImplementedError。通过引入 CpuDeviceMixin 和 CpuSRTPlatform，使三个主要内置设备都有具体的平台类返回，避免 CPU 落在抽象基类上。

实现拆解

1. 新建 CPU 平台实现：文件 python/sglang/srt/platforms/cpu.py，包含 CpuDeviceMixin(DeviceMixin) 和 CpuSRTPlatform(CpuDeviceMixin, SRTPlatform)。CpuDeviceMixin 实现了 DeviceMixin 定义的全部 12 个方法，包括设备标识、内存查询（psutil）、empty_cache（触发 gc.collect()）、synchronize（调用 torch.cpu.synchronize()）等。CpuSRTPlatform 覆写了功能开关（supports_fp8 等返回 False）和 is_pin_memory_available（返回 False）。
2. 更新平台发现逻辑：修改 python/sglang/srt/platforms/__init__.py，新增 _is_cpu_available() 检查环境变量 SGLANG_USE_CPU_ENGINE。在 _resolve_platform() 的自动发现分支中，当无插件激活时，首先检查 CPU 是否启用（优先生效），再检查 CUDA / ROCm，最后回退到抽象基类。该顺序允许开发者在 GPU 主机上显式测试 CPU 路径。
3. 增加单元测试：在 test/registered/unit/platforms/test_platform_interface.py 中添加 TestCpuDeviceMixin 类（14 个测试），覆盖平台身份、设备返回、内存查询（psutil mock）、empty_cache 调用 gc.collect、synchronize 调用、分布式后端 gloo、架构分支（ARM/x86）、能力标志等。同时新增 3 个平台解析器测试（CPU 启用 / 未启用 / 同时有 CUDA 时的优先级）。

4. 同步文档：更新 docs_new/docs/hardware-platforms/plugin.mdx 中的平台发现流程图，加入 CPU 回退路径，并同步已有 CUDA/ROCm 回退路径以反映实际 _resolve_platform() 逻辑。

关键文件：

- python/sglang/srt/platforms/cpu.py（模块 平台层；类别 source；类型 core-logic；符号 CpuDeviceMixin, CpuSRTPlatform, cpu_arch, get_device_total_memory）：核心新增文件：定义 CpuDeviceMixin 和 CpuSRTPlatform，实现 CPU 设备操作和平台类
- python/sglang/srt/platforms/__init__.py（模块 平台层；类别 source；类型 core-logic；符号 _is_cpu_available）：修改平台发现逻辑：新增 _is_cpu_available，在 _resolve_platform 中优先检查 CPU 回退
- test/registered/unit/platforms/test_platform_interface.py（模块 平台测试；类别 test；类型 test-coverage；符号 TestCpuDeviceMixin, test_cpu_platform_identity, test_default_get_device_returns_cpu_device, test_default_get_device_total_memory_uses_psutil）：新增 TestCpuDeviceMixin 类（14 测试）和 3 个平台解析测试，确保 CPU 平台类行为正确
- docs_new/docs/hardware-platforms/plugin.mdx（模块 文档；类别 docs；类型 documentation）：更新平台发现流程图，加入 CPU 回退路径并同步 CUDA/ROCm 路径

关键符号：CpuDeviceMixin.get_device_total_memory, CpuDeviceMixin.get_current_memory_usage, CpuDeviceMixin.get_device, CpuDeviceMixin.set_device, CpuDeviceMixin.get_device_name, CpuDeviceMixin.get_device_uuid, CpuDeviceMixin.empty_cache, CpuDeviceMixin.synchronize, CpuDeviceMixin.cpu_arch, CpuSRTPlatform.is_pin_memory_available, _is_cpu_available, _resolve_platform

关键源码片段

python/sglang/srt/platforms/cpu.py

核心新增文件：定义 CpuDeviceMixin 和 CpuSRTPlatform，实现 CPU 设备操作和平台类

```
import gc
import platform as _platform
from functools import cached_property
from typing import Optional

import psutil
import torch

from sglang.srt.platforms.device_mixin import (
    CpuArchEnum,
    DeviceCapability,
    DeviceMixin,
    PlatformEnum,
)
from sglang.srt.platforms.interface import SRTPlatform
```

```
class CpuDeviceMixin(DeviceMixin):
    """CPU 设备操作混合类，实现 DeviceMixin 抽象接口。"""
    _enum: PlatformEnum = PlatformEnum.CPU
    device_name: str = "cpu"
    device_type: str = "cpu"

    @cached_property
    def cpu_arch(self) -> CpuArchEnum:
        """主机 CPU 架构（X86 / ARM / UNSPECIFIED），进程内只解析一次。"""
        return self.get_cpu_architecture()

    def get_device_total_memory(self, device_id: int = 0) -> int:
        # 返回整机物理内存总量（字节），与 PSUTIL 一致
        return int(psutil.virtual_memory().total)

    def get_current_memory_usage(
        self, device: Optional["torch.device"] = None
    ) -> float:
        """整机已用内存（total - available），而非进程 RSS。
        该方法遵循 [Active] 契约：free = total - used 应当正确反映
        系统可用内存（psutil.available）。返回浮点字节数。
        """
        vm = psutil.virtual_memory()
        return float(vm.total - vm.available)

    def get_device(self, local_rank: int) -> "torch.device":
        # CPU 只有一个设备，忽略 local_rank；rank 隔离通过 numactl / OpenMP 绑定实现
        # TODO(zijiexia): 后续可支持 NUMA 感知的 rank 放置
        return torch.device("cpu")

    def set_device(self, device: "torch.device") -> None:
        # CPU 上为显式无操作；避免使用 torch.set_default_device("cpu") 改变全局默认张量设备
        torch.cpu.set_device(device)

    def get_device_name(self, device_id: int = 0) -> str:
        # 返回基于架构的简短描述，避免调用 platform.processor()（可能产生子进程）
        if self.cpu_arch == CpuArchEnum.ARM:
            return "cpu (aarch64)"
        if self.cpu_arch == CpuArchEnum.X86:
            return "cpu (x86_64)"
        return "cpu"

    def get_device_uuid(self, device_id: int = 0) -> str:
        # CPU 无设备 UUID，以平台架构字符串作为稳定主机标识
        return _platform.machine()

    def get_device_capability(self, device_id: int = 0) -> Optional[DeviceCapability]:
        return None
```

```
    def empty_cache(self) -> None:
        # CPU 上无 torch.cpu.empty_cache()，通过 GC 回收引用循环内存。
        # 注意：gc.collect() 的暂停时间随堆大小增长，且释放的内存在 tcmalloc /
        # TBB malloc 下不会返还给 OS，需后续通过 allocator 感知调用改进
        gc.collect()

    def synchronize(self) -> None:
        # CPU 无异步流，调用 torch.cpu.synchronize() 保持与 CudaDeviceMixin 的对称性
        torch.cpu.synchronize()
```

评论区精华

- 性能优化建议：gemini-code-assist[bot] 建议缓存 psutil.Process() 实例以避免每次调用 get_current_memory_usage 的开销；建议在 empty_cache 中通过 ctypes 调用 malloc_trim 释放堆内存回 OS。开发者 zijiexia 回应指出 CPU 部署指南预加载 tcmalloc/TBB malloc，malloc_trim 无效，且 gc.collect() 已在基类中定义为 no-op，当前实现已比原基类更主动。alexnails 同意添加注释说明潜在问题。
- 架构标识设计：alexnails 要求将 CPU 架构作为一等属性体现。zijiexia 新增了 cpu_arch cached_property，从 get_cpu_architecture() 解析，用于 get_device_name() 输出架构标签。
- get_device 的 NUMA 感知：alexnails 指出 CPU 上可以有多个 rank，建议后续支持 NUMA 感知的 rank 放置。zijiexia 添加了 TODO 注释。
- empty_cache 的 allocator 局限：讨论确认当前 gc.collect() 在 tcmalloc/TBB malloc 下不会将内存返还 OS，但该问题属于低频清理路径，优先级低。
  - empty_cache 实现与 malloc_trim 讨论 (performance): 保留 gc.collect()，添加注释说明在 tcmalloc/TBB malloc 下 RSS 不降，未来可通过 allocator 感知改进。
  - get_device 的 NUMA 感知 (design): 添加 TODO 注释，后续可支持 NUMA 感知的 rank 放置。
  - CPU 架构作为一等属性 (design): 新增 cpu_arch 属性，避免重复调用 platform.machine()。

风险与影响

- 风险：
  - psutil 调用性能风险：get_device_total_memory 和 get_current_memory_usage 每次调用都执行系统级查询（psutil.virtual_memory()），若被热路径频繁调用可能引入额外开销。当前 PR 未迁移任何运行时调用，风险仅存在于未来启用后。
  - gc.collect() 暂停：在空缓存、空闲休眠时调用 gc.collect() 可能因堆大小引入毫秒级暂停，但出现在清理路径，可接受。未使用 malloc_trim 可能导致 RSS 不降，但符合 CPU 指导（预加载 tcmalloc）。
  - 环境变量竞争：SGLANG_USE_CPU_ENGINE=1 在 GPU 主机上会覆盖 CUDA/ROCm 回退，若用户无意设置可能导致异常行为。但该变量为显式 opt-in，风险可控。

    - 平台发现顺序改变：CPU 回退现在优先于 CUDA/ROCm，若未来有同等优先级的其他平台（如 XPU/HPU），可能需要调整排序逻辑。当前仅 CPU 使用环境变量 opt-in，无实际冲突。
- 影响：
    - 用户影响：对使用 CPU 引擎的用户（设置 SGLANG_USE_CPU_ENGINE=1）透明，现在 current_platform 返回 CpuSRTPlatform 而非抽象基类，empty_cache() 从 pass 提升为 gc.collect()，内存查询从报错变为返回真实值。对其他用户无影响。
    - 系统影响：新增 cpu.py 和 __init__.py 中的平台发现逻辑，测试套件扩展。无运行时性能影响，所有变更均为静态平台类定义和测试。
    - 团队影响：为后续迁移分布式后端（get_torch_distributed_backend_str → gloo）、内存管理（is_pin_memory_available → False）等基础设施提供基础。其他设备（XPU/HPU/NPU/MUSA）可参考此模式添加自己的平台类。
    - 风险标记：psutil 调用可能引入热路径开销，gc.collect() 暂停时间随堆增长，环境变量意外设置导致 GPU 主机使用 CPU 路径，平台发现顺序变更可能影响未来多平台排序

关联脉络

- PR #24096 Introduce CudaDeviceMixin and CudaSRTPlatform (and ROCm variants): 该 PR 是 #24096 的镜像，为 CPU 添加对应的平台类，设计模式完全一致。