

PR #39516 完整报告

sgl-project/sglang

[Fix] HiCache startup ImportError on the pinned kernel wheel

合并时间: 2026-09-15 13:31

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39516>

执行摘要

- 一句话: 修复 HiCache 启动时 ImportError 崩溃
- 推荐动作: 值得精读, 但属于中等篇幅即可读完的 PR (3 文件、+83/-5)。重点看三处:
 - (1) `_resolve_device_accessible_ptr_fn` 的双重存在性检查 (`import + torch.ops`) 与 `lru_cache(maxsize=1)` 的收敛设计, 这是跨仓库发布顺序错配时的通用处置模式;
 - (2) HIP 上 fail-fast、CUDA 上 fail-soft 的平台分工, 背后的依据是 UVA 下 host/device 地址别名关系;
 - (3) 测试如何用 `alloc_with_host_register + monkeypatch resolver` 直接验证回退等价性, 以及为什么在缺 op 时选择 skip 而不是 fail。阅读时请额外确认夹带的 `unified_radix_cache.py` 选择器改动 (`req_id → request.rid`) 在本仓库当前签名下确实必要, 这行改动没有配套测试。若你是部署 /CI 维护者, 可作为「AOT op 与调用方必须分两个 PR 落地」的案例收藏。

功能与动机

根因是发布顺序缺失: PR body 明确写到 #35233 把 AOT op 和调用方放在一个 PR 里落地, 跳过了贡献指南 Update sglang-kernel 一节要求的顺序。PR CI 构建 PR-local wheel, 所以当时是绿的; 而 main 与定时 CI 跑的是 pinned sglang-kernel==0.4.7, 那里没有这个符号, 于是「any server that builds a registered host pool dies at startup on ImportError: cannot import name 'get_device_accessible_ptr'. That was 32 jobs in the last scheduled run.」作者同时排除了直接回滚: ROCm 从本仓库构建 kernel、已经在跑这个修复, 回滚会重新打断 MI355X, 而且这个算子终究要发布。因此采取指南允许的例外——先做存在性检查, 保留等价回退, 直到已发布 wheel 追上来。Issue 评论中 HaiShaw 对 AMD 同学补充了「FYI. Let's do more JIT, and be careful with AOT process.」, 把这次事故归因到 AOT 流程纪律上。

实现拆解

1. 新增可测试、可缓存的算子解析入口 —— `python/sglang/srt/mem_cache/pool_host/common.py` - 新增 `from functools import lru_cache`、`from sglang.srt.utils import is_hip`, 并在模块层缓存 `_is_hip = is_hip()`。 - 新增 `_resolve_device_accessible_ptr_fn()` 并加 `@lru_cache(maxsize=1)`: 进程内只解析一次, 把「op 是否存在」的判断从调用点内联 `import` 收敛成单一入口, 便于测试 `monkeypatch`, 也便于将来 pin 更新后一处收口。 - 双重存在性检查: `from sgl_kernel.kvcacheio import get_device_accessible_ptr` 捕获 `ImportError` 置 `None`; `import` 成功后再用 `hasattr(torch.ops.sgl_kernel, "`

get_device_accessible_ptr") 覆盖「符号在但算子未注册」的情形。

2. 平台差异化的降级策略 —— 同一文件 - HIP (`_is_hip` 为真) : 直接 `raise ImportError`, 错误信息给出重建指引 `python/sglang/kernels/aot (setup_rocm.py)`。因为 ROCm 上 `registered host memory` 携带独立的 `device` 地址, 静默回退会写错指针, 缺 `op` 即意味着构建是坏的。 - CUDA: 只 `logger.warning` 一次并返回 `None`, 让调用方走原始地址。因为 UVA 下 `host` 地址本身就是 `device alias`, 「nothing changes past one warning」。
3. 调用方改造 `make_kernel_ptr_table` —— 同一文件 - 原实现在 `host_memory_registered and device.type == "cuda"` 分支里内联 `from sglang.kvcacheio import ...`; 现在先调用 `_resolve_device_accessible_ptr_fn()`, 返回 `None` 就落到 `tensor.data_ptr()` 分支。 - 效果: 缺少 `op` 的 CUDA wheel 上, 行为回到 #35233 之前的语义, 不会在启动 / 建表阶段抛异常; 地址计算仍是 `torch.tensor(pointers, dtype=torch.uint64, device=device)`, 其余逻辑不动。
4. 测试配套 —— `test/registered/kernels/ops/kvcache/test_hicache_page_first_write_back.py` - 新增 `test_registered_mmap_kernel_ptr_table_fallback_matches_device_alias`: HIP 上 `skipif`; `resolver` 返回 `None` 时 `pytest.skip`; 否则用 `alloc_with_host_register((PAGE_SIZE * 4, 128), torch.bfloat16, "cpu", True, HostTensorAllocator())` 建真实 `pinned` 内存, 比较别名表与 `monkeypatch` 掉 `resolver` 后的 `raw` 表, `finally` 中 `_cuda_host_unregister` 释放。 - 已有 `test_registered_mmap_pointer_domains_and_all_layer_transfer` 由「直接 `import`」改为走 `_resolve_device_accessible_ptr_fn()`, 拿不到就 `skip` 并提示从 AOT 构建, 避免 `pinned wheel` 下套件整体变红。
5. 顺带修复 —— `python/sglang/srt/mem_cache/unified_radix_cache.py` - `prefetch_from_storage` 的 `@rank_consensus(same_params=["req_id", "len(new_input_tokens)])` 改为 `same_params=["request.rid", "len(new_input_tokens)"]`, 与实参 `request: CacheRequestHandle` 对齐; 该改动来自维护者 `hnyls2002` 的 `commit` 「fix prefetch consensus selector」, 说明它是合入过程中被发现并顺手修掉的选择器失配, 而不是本 PR 的主线目标。

文件	类别	关键符号	作用
python/sglang/srt/mem_cache/pool_host/common.py	source	<code>_resolve_device_accessible_ptr_fn</code> 、 <code>make_kernel_ptr_table</code>	主线修复: 解析 + 平台差异化降级
test/registered/kernels/ops/kvcache/test_hicache_page_first_write_back.py	test	<code>test_registered_mmap_kernel_ptr_table_fallback_matches_device_alias</code>	校验 CUDA 回退与别名表等价, 并让原测试在缺 <code>op</code> 时跳过

文件	类别	关键符号	作用
python/sglang/srt/mem_cache/unified_radix_cache.py	source	<code>prefetch_from_storage</code>	顺带修正 rank_consensus 选择器字段

关键文件：

- python/sglang/srt/mem_cache/pool_host/common.py（模块 主机内存池；类别 source；类型 dependency-wiring；符号 `_resolve_device_accessible_ptr_fn`, `make_kernel_ptr_table`, `is_hip`）：主线修复文件：新增 `_resolve_device_accessible_ptr_fn()` 承担算子存在性检查与平台差异化降级，并改造 `make_kernel_ptr_table` 的指针表构建路径，直接决定 HiCache registered host pool 能否在 pinned wheel 上启动。
- test/registered/kernels/ops/kvcache/test_hicache_page_first_write_back.py（模块 内核测试；类别 test；类型 test-coverage；符号 `test_registered_mmap_kernel_ptr_table_fallback_matches_device_alias`, `test_registered_mmap_pointer_domains_and_all_layer_transfer`）：测试配套：新增 CUDA 回退等价性测试并用 monkeypatch 覆盖回退分支，同时把既有 registered 内存指针测试改为走 resolver 并在缺 op 时跳过，保证 pinned wheel 下套件不会整体变红。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 前缀缓存；类别 source；类型 core-logic；符号 `prefetch_from_storage`）：顺带修复：`prefetch_from_storage` 的 `rank_consensus` 选择器由 `req_id` 改为 `request.rid`，与实参名对齐，属合入过程中被维护者发现的选择器失配。

关键符号：`_resolve_device_accessible_ptr_fn`, `make_kernel_ptr_table`, `test_registered_mmap_kernel_ptr_table_fallback_matches_device_alias`, `test_registered_mmap_pointer_domains_and_all_layer_transfer`, `prefetch_from_storage`

关键源码片段

python/sglang/srt/mem_cache/pool_host/common.py

主线修复文件：新增 `_resolve_device_accessible_ptr_fn()` 承担算子存在性检查与平台差异化降级，并改造 `make_kernel_ptr_table` 的指针表构建路径，直接决定 HiCache registered host pool 能否在 pinned wheel 上启动。

```
# python/sglang/srt/mem_cache/pool_host/common.py
from functools import lru_cache

from sglang.srt.utils import is_hip

# 平台判定只在模块加载时做一次，避免在解析函数里重复调用 is_hip()
```

```
_is_hip = is_hip()
```

```
@lru_cache(maxsize=1)
```

```
def _resolve_device_accessible_ptr_fn():
```

```
    """解析 sgl_kernel 的 device-accessible 指针算子；缺失时按平台降级或报错。
```

```
    maxsize=1 的缓存确保每个进程只解析一次：wheel 内容在进程内不会变化。
```

```
    """
```

```
    try:
```

```
        # 第一层：pinned wheel 里可能没有这个模块级符号（#35233 新增的 AOT op）
```

```
        from sgl_kernel.kvcacheio import get_device_accessible_ptr
```

```
    except ImportError:
```

```
        get_device_accessible_ptr = None
```

```
    else:
```

```
        # 第二层：符号存在但 torch.ops 命名空间未注册时，调用依然会失败
```

```
        if not hasattr(torch.ops.sgl_kernel, "get_device_accessible_ptr"):
```

```
            get_device_accessible_ptr = None
```

```
    if get_device_accessible_ptr is None:
```

```
        # CUDA 的 UVA 让 host 地址本身就是 device 可见地址，回退等价；
```

```
        # ROCm 上两个地址确实不同，静默回退会写错指针，因此直接 fail fast。
```

```
        if _is_hip:
```

```
            raise ImportError(
```

```
                "sgl_kernel.kvcacheio.get_device_accessible_ptr is missing from the "
```

```
                "installed sglang-kernel. It is required on ROCm, where registered "
```

```
                "host memory carries a distinct device address. Rebuild sglang-kernel "
```

```
                "from python/sglang/kernels/aot (setup_rocm.py)."
```

```
            )
```

```
        logger.warning(
```

```
            "sgl_kernel.kvcacheio.get_device_accessible_ptr is missing from the "
```

```
            "installed sglang-kernel; using raw host addresses for kernel pointer "
```

```
            "tables. Build sglang-kernel from python/sglang/kernels/aot to enable it."
```

```
        )
```

```
    return get_device_accessible_ptr
```

```
def make_kernel_ptr_table(
```

```
    tensors: list[torch.Tensor],
```

```
    target_device: torch.device | str,
```

```
    *,
```

```
    host_memory_registered: bool,
```

```
) -> torch.Tensor:
```

```
    # 只有 registered 的 CUDA host 内存才需要地址转换，其他情况一律用原始地址
```

```
    device = torch.device(target_device)
```

```
    get_device_accessible_ptr = (
```

```
        _resolve_device_accessible_ptr_fn()
```

```
        if host_memory_registered and device.type == "cuda"
```

```
        else None
```

```

)
if get_device_accessible_ptr is not None:
    if device.index is None:
        device_index = torch.cuda.current_device()
    else:
        device_index = device.index
    pointers = [
        get_device_accessible_ptr(tensor, device_index) for tensor in tensors
    ]
else:
    # 回退分支: 缺 op 的 CUDA wheel 上, UVA 让 data_ptr() 即为 device 别名
    pointers = [tensor.data_ptr() for tensor in tensors]
return torch.tensor(
    pointers,
    dtype=torch.uint64,
    device=device,
)

```

test/registered/kernels/ops/kvcache/test_hicache_page_first_write_back.py

测试配套: 新增 CUDA 回退等价性测试并用 monkeypatch 覆盖回退分支, 同时把既有 registered 内存指针测试改为走 resolver 并在缺 op 时跳过, 保证 pinned wheel 下套件不会整体变红。

```

# test/registered/kernels/ops/kvcache/test_hicache_page_first_write_back.py
@pytest.mark.skipif(
    is_hip(),
    reason="ROCm maps registered host memory at a distinct device address.",
)
def test_registered_mmap_kernel_ptr_table_fallback_matches_device_alias(
    monkeypatch: pytest.MonkeyPatch,
) -> None:
    """CUDA maps registered host memory at the host address itself;
    ``make_kernel_ptr_table``'s raw-host-address fallback depends on it."""
    # pinned wheel 没有该 op 时别名表本身无法构建, 测试跳过而不是失败
    if _resolve_device_accessible_ptr_fn() is None:
        pytest.skip(
            "installed sglang-kernel has no get_device_accessible_ptr; "
            "build it from python/sglang/kernels/aot to run this test"
        )

    # 使用真实注册的 pinned host 内存, 避免 UVA 别名关系是伪造出来的
    buffer = alloc_with_host_register(
        (PAGE_SIZE * 4, 128),
        torch.bfloat16,
        "cpu",
        True,
        HostTensorAllocator(),
    )
    try:

```

```

aliased = make_kernel_ptr_table([buffer], DEVICE, host_memory_registered=True)
# 强行让 resolver 返回 None, 强制走 raw data_ptr() 回退路径
monkeypatch.setattr(
    pool_host_common, "_resolve_device_accessible_ptr_fn", lambda: None
)
raw = make_kernel_ptr_table([buffer], DEVICE, host_memory_registered=True)
# 两者不等说明 UVA 回退假设被打破, 回退路径会写错指针
assert torch.equal(aliased, raw)
finally:
    _cuda_host_unregister(buffer)

```

```

def test_registered_mmap_pointer_domains_and_all_layer_transfer() -> None:
    # 统一走 resolver: 缺 op 的 wheel 上跳过, 而不是在 import 阶段就 ImportError
    get_device_accessible_ptr = _resolve_device_accessible_ptr_fn()
    if get_device_accessible_ptr is None:
        pytest.skip(
            "installed sglang-kernel has no get_device_accessible_ptr; "
            "build it from python/sglang/kernels/aot to run this test"
        )

```

python/sglang/srt/mem_cache/unified_radix_cache.py

顺带修复: prefetch_from_storage 的 rank_consensus 选择器由 req_id 改为 request.rid, 与实参名对齐, 属合入过程中被维护者发现的选择器失配。

```

# python/sglang/srt/mem_cache/unified_radix_cache.py
# rank_consensus 的 same_params 选择器此前写的是 "req_id", 与实参名 request 不匹配,
# 会让多 rank 的 prefetch 共识校验取不到值; 此处顺带改成 request.rid。
@rank_consensus(same_params=["request.rid", "len(new_input_tokens)"])
def prefetch_from_storage(
    self,
    request: CacheRequestHandle,
    # 其余参数 last_host_node_id / new_input_tokens / last_hash / prefix_keys /
    # matched_prefix_tokens / extra_key / cache_salt 以及函数体均未改动

```

评论区精华

Review 侧只有一条 LGTM, 实质论证集中在 PR body 与 commit 记录里:

HaiShaw (APPROVED) : “LGTM” HaiShaw (Issue 评论, @AMD-yanfeiwang @amd-danli103 @1am9trash @zijiecode) : “FYI. Let's do more JIT, and be careful with AOT process.”

PR body 主动交代了取舍: “That was 32 jobs in the last scheduled run. Reverting re-breaks MI355X ... So this takes the exception that section allows: check the op is there, keep an equivalent fallback until a released wheel catches up.” 并明确了 CUDA 回退等价 (UVA 即 device alias)、ROCm 必须 raise 的平台分工。

commits 显示维护者 hnyls2002 在合入前做了最后打磨 (“trim hicache ptr fallback comments”), 并亲自提交 “fix prefetch consensus selector”, 把 `req_id` 选择器改成 `request.rid`。值得注意的是: 没有任何逐行 review 评论, 也没有人讨论“长期停留在回退分支会不会掩盖 wheel 未更新”这类 observability 问题, 回退的去留被留给了将来的 pin 更新 (“the fallback can go or stay as skew insurance”)。

- AOT 算子与调用方向 PR 合入导致 pinned wheel 版本错配 (design): 采取指南允许的例外: 先检查 op 是否存在, 保留等价回退等待已发布 wheel 追上, 待 pin 更新后回退可移除或保留为版本错配保险。
- CUDA 回退与 ROCm fail-fast 的平台分工是否合理 (design): 平台差异化策略被接受, HaiShaw 以 APPROVED + LGTM 放行; 维护者仅做了注释精简 (commit “trim hicache ptr fallback comments”)。
- 缺 op 环境下测试该跳过还是失败 (testing): 选择 skip 以保证 pinned wheel 下套件保持绿色, 代价是这些断言在旧 wheel 环境下并不真正执行, 等 pin 更新后才会实际运行。
- 顺带修正 rank_consensus 选择器 (`req_id` → `request.rid`) (correctness): 已随本 PR 合入, 但没有独立测试、也没有对应讨论记录, 属于被裹挟进修复 PR 的小范围正确性改动。

风险与影响

- 风险:
 1. 回退路径依赖 CUDA UVA 语义: `make_kernel_ptr_table` 在缺 op 时用 `tensor.data_ptr()` 作为 device 可见地址, 仅在 UVA 下成立; 新增测试在 CUDA 上对此做了断言 (`aliased == raw`), 但在非 UVA 或未来引入不同 pinned 内存注册方式时, 该假设需要重新验证。
 2. `@lru_cache(maxsize=1)` 把解析结果固化在进程生命周期内: 一旦在缺 op 的 wheel 上启动, 即便后续 AOT 构建补齐也不会热生效, 必须重启进程; 这属于可接受的取舍, 但排障时容易被忽略 (尤其只有一条 warning)。
 3. ROCm 是刻意的 fail-fast: 缺少 op 时 `raise ImportError`, 把失败从 import 期推后到构建 registered host pool 时, 错误信息更明确, 但使用 pinned wheel 的 ROCm 集群会直接启动失败——这正是设计意图, 不属于 bug, 但会带来部署侧的硬性要求 (必须从 `python/sglang/kernels/aot` 构建)。
 4. 测试覆盖有缺口: 新测试只覆盖 CUDA 回退分支, ROCm 的 `raise` 分支和 `torch.ops` 二次检查都没有直接测试; 此外缺 op 时测试走 `pytest.skip`, 意味着在 pinned wheel 环境下这些断言实际上不会执行, 回归保护存在真空期。
 5. `unified_radix_cache.py` 的 `rank_consensus(same_params=["request.rid", ...])` 属夹带修复: 若 `CacheRequestHandle` 上没有 `rid` 字段或语义不符, 多 rank 的 prefetch 共识校验会静默失效, 而本 PR 没有为这行改动补测试。
 6. 回退在 CUDA 上只打一次 warning, 缺少指标 / 计数暴露; 长期跑在回退路径上不会在监控里体现。- 影响: 用户侧: 使用官方 pinned `sglang-kernel wheel (0.4.7)` 的 CUDA 用户, 此前只要开启 registered host pool (`HiCache page_first + kernel` 模式的写回 / 加载路径) 就会在启动时 `ImportError` 崩溃, 现在可正常启动, 代价仅是日志里一条 warning; ROCm 用户不受影响, 仍要求从源码构建 kernel (否则明确报错而不是算错指针)。系统与 CI 侧: 恢复 main 与定时 CI 的绿色 (此前 32 个 job 因该

ImportError 失败)，并让 PR CI 与 main CI 的行为差异（PR-local wheel vs pinnedwheel）显式化。团队流程侧：本 PR 是「AOT 算子需先发布 wheel、再合入调用方」这条纪律缺口的直接产物，Issue 评论中 AMD 相关同学被点名提醒“do more JIT, be careful with AOT process”，预计会推动后续同类改动优先走 JIT 路径或补上 fallback。影响范围集中在 HiCache host 内存池的建表入口，不触及正常 AOT wheel 的服务路径。- 风险标记：启动路径降级，依赖 CUDA UVA 平台假设，ROCm 缺 op 直接 raise，缺 op 时测试跳过，夹带 rank_consensus 修复，lru_cache 固化解析结果

关联脉络

- PR #35233 [AMD] Fix registered HiCache host pointer aliases: 直接上游：该 PR 同时落地 AOT 算子 get_device_accessible_ptr 与调用方 make_kernel_ptr_table（含同一测试文件与 pool_host/common.py），跳过 sglang-kernel 发布顺序，是本 PR 要修的全部原因；正文明确说明回滚它会重新打断 MI355X。
- PR #39318 Scope prefetch cache state to the request attempt: 同样改动 python/sglang/srt/mem_cache/unified_radix_cache.py 与 prefetch/ 请求命名空间逻辑，很可能是 prefetch_from_storage 实参被称为 request 的来源；本 PR 顺带修正的 rank_consensus 选择器与之属同一功能线（需人工确认因果关系）。