PR #36705 完整报告

sgl-project/sglang

[HiCache] Stop populating host-pool mmaps twice (-13% allocation time)

合并时间：2026-08-28 11:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36705

执行摘要

- 一句话：去重 host-pool mmap 双重预填充，分配耗时降约 13%
- 推荐动作：值得精读。该 PR 展示了系统调用能力探测 + 回退的稳妥写法，用 mincore 直接验证不变式而非从 flags 反推，并通过 mock 强制覆盖 CI 上不可达的回退分支，避免未测试代码合入；同时修复了异常吞噬问题。对从事启动性能、共享内存分配或 HiCache 相关工作的同学有直接参考价值。

功能与动机

Issue #31424 指出 HiCache host buffer 分配开销大，且原实现存在双重预填充：mmap 时传 MAP_POPULATE，随后又对同一范围调用 madvise(MADV_POPULATE_WRITE)。由于 cudaHostRegister 会将这些页面钉住，任一方式单独即可保证“返回时页面已 fault-in 且可写”，同时使用只会让内核额外走一遍全量映射。HiCache 场景下 --hicache-ratio 6 且 PP=2 时每个 rank 分配 418 GB，且该分配位于 server 启动关键路径，浪费明显。此外旧代码用 except OSError: pass 吞掉 madvise 失败，会把 ENOMEM 也一起忽略，与其注释宣称的“失败时抛错（如内存不足）”自相矛盾。

实现拆解

1. 在 python/sglang/srt/mem_cache/storage/mmap/mmap_allocator.py 中新增 _has_madv_populate_write()，用 functools.cache 缓存探测结果；探测基于单页 MAP_PRIVATE 匿名映射调用 madvise(MADV_POPULATE_WRITE) 是否成功，成功即视为内核支持（Linux 5.14+）。
2. 新增 _mmap_prefaulted(fileno, alloc_bytes, flags) 统一封装：支持 madvise 时先 mmap（不带 MAP_POPULATE）再 madvise；不支持时回退 flags | _MAP_POPULATE。两条路径都必须保证返回时映射已全页 fault-in 且可写，以满足 cudaHostRegister 的钉页前提。
3. alloc_mmap 与 alloc_shm 的普通路径改为调用 _mmap_prefaulted，删除原先“MAP_POPULATE + madvise”的组合以及 except OSError: pass 的异常吞噬逻辑；madvise 失败（如 ENOMEM）现在直接向上传播。hugepage 路径 _alloc_hugepage 保持原逻辑，因为 MAP_HUGETLB 已提前预留页。
4. 测试侧：test/registered/unit/mem_cache/test_mmap_allocator.py 新增 test_mmap_prefaulted_leaves_no_lazy_page，用 mincore() 逐页断言两条路径都返回 fully resident 的映射；由于 CI 内核（5.14+）上 MAP_POPULATE 分支不可达，测试通过 unittest.mock.patch 强制探针返回 False 来覆盖该分支，避免未测试代码合入。

5. 实测数据：作者在 288 核 / 3 TB 主机、kernel 6.8 上对 8/16/32/64 GiB 分配测得耗时下降 13.0%-13.6%，mincore 驻留率前后均为 1.0；另在 5.15 内核做了 mutation 测试，确保 madvise 丢失或回退 flag 丢失都会使对应子测试失败。

关键文件：

- python/sglang/srt/mem_cache/storage/mmap/mmap_allocator.py（模块 内存映射；类别 source；类型 core-logic；符号 _has_madv_populate_write, _mmap_prefaulted, alloc_mmap, alloc_shm）：核心变更文件：新增内核能力探针 _has_madv_populate_write 与统一预填充封装 _mmap_prefaulted，alloc_mmap/alloc_shm 改为调用封装，去掉 MAP_POPULATE + madvise 双重预填充，并让 madvise 失败（如 ENOMEM）直接抛出。
- test/registered/unit/mem_cache/test_mmap_allocator.py（模块 分配测试；类别 test；类型 test-coverage；符号 _assert_resident, test_mmap_prefaulted_leaves_no_lazy_page）：测试配套：新增 test_mmap_prefaulted_leaves_no_lazy_page，通过 mincore() 逐页断言两条预填充路径都返回完全驻留的映射；并用 unittest.mock.patch 强制探针返回 False，覆盖 CI 内核上不可达的 MAP_POPULATE 回退分支。

关键符号：alloc_mmap, alloc_shm, _has_madv_populate_write, _mmap_prefaulted, _assert_resident, test_mmap_prefaulted_leaves_no_lazy_page

关键源码片段

python/sglang/srt/mem_cache/storage/mmap/mmap_allocator.py

核心变更文件：新增内核能力探针 _has_madv_populate_write 与统一预填充封装 _mmap_prefaulted，alloc_mmap/alloc_shm 改为调用封装，去掉 MAP_POPULATE + madvise 双重预填充，并让 madvise 失败（如 ENOMEM）直接抛出。

```
# python/sglang/srt/mem_cache/storage/mmap/mmap_allocator.py（整理后）

# 探测内核是否实现 MADV_POPULATE_WRITE（Linux 5.14+）。
# 只在单页上探测一次，结果被 functools.cache 缓存供所有分配复用。
@functools.cache
def _has_madv_populate_write() -> bool:
    """Whether this kernel implements MADV_POPULATE_WRITE (Linux 5.14+)."""
    try:
        # 用 MAP_PRIVATE 匿名映射一个 PAGESIZE 的探测页，避免影响真实分配
        probe = mmap.mmap(
            -1,
            mmap.PAGESIZE,
            flags=mmap.MAP_PRIVATE | mmap.MAP_ANONYMOUS,
            prot=_PROT_RW,
        )
    except OSError:
        return False
    try:
        # madvise 成功即视为内核支持该操作
        probe.madvise(_MADV_POPULATE_WRITE)
```

```
            return True
        except (OSError, ValueError):
            return False
        finally:
            probe.close()


    # 返回一个每页都已 fault-in 且可写的 mmap。
    # cudaHostRegister 必须钉住真实、已预填充的物理页，因此这类映射
    # 绝不能延迟交付。MAP_POPULATE 与 MADV_POPULATE_WRITE 各自都能
    # 提供该保证，但两者同时使用会令内核遍历整个映射两次。优先使用
    # madvise（它还能把 ENOMEM 之类的真实失败上报，而不是静默留下
    # 未填充页），仅在缺少该能力的内核上回退到 MAP_POPULATE。
    def _mmap_prefaulted(fileno: int, alloc_bytes: int, flags: int) -> mmap.mmap:
        if _has_madv_populate_write():
            # Linux 5.14+：先 mmap 再 madvise，一次遍历完成预填充
            mm = mmap.mmap(fileno, alloc_bytes, flags=flags, prot=_PROT_RW)
            # 出错时直接抛出，不再被吞掉（旧实现 except OSError: pass 会掩盖 ENOMEM）
            mm.madvise(_MADV_POPULATE_WRITE)
            return mm
        # 旧内核回退：靠 MAP_POPULATE 在 mmap 时同步预填充
        return mmap.mmap(fileno, alloc_bytes, flags=flags | _MAP_POPULATE, prot=_PROT_RW)
```

test/registered/unit/mem_cache/test_mmap_allocator.py

测试配套：新增 test_mmap_prefaulted_leaves_no_lazy_page，通过 mincore() 逐页断言两条预填充路径都返回完全驻留的映射；并用 unittest.mock.patch 强制探针返回 False，覆盖 CI 内核上不可达的 MAP_POPULATE 回退分支。

```
    # test/registered/unit/mem_cache/test_mmap_allocator.py（整理后）

    # 用 mincore() 逐页检查映射是否全部驻留内存。
    # 直接核验“每页都在”这一不变式，而不是从传入的 flags 反推。
    def _assert_resident(self, mm, alloc_bytes):
        addr = ctypes.addressof(ctypes.c_char.from_buffer(mm))
        npages = alloc_bytes // mmap.PAGESIZE
        vec = (ctypes.c_ubyte * npages)()
        libc = ctypes.CDLL(ctypes.util.find_library("c"), use_errno=True)
        if libc.mincore(ctypes.c_void_p(addr), ctypes.c_size_t(alloc_bytes), vec) != 0:
            self.skipTest("mincore unavailable")
        self.assertTrue(all(v & 1 for v in vec), "mapping was not fully pre-faulted")


    # 两条预填充路径都必须返回完全驻留的映射。
    # cudaHostRegister 会钉住这些缓冲区；如果注册时还有页未 fault-in，
    # 设备可能读到尚未分配后备存储的旧数据。
    def test_mmap_prefaulted_leaves_no_lazy_page(self):
        alloc_bytes = 64 * mmap.PAGESIZE
        flags = mmap.MAP_SHARED | mmap.MAP_ANONYMOUS
```

```
        with self.subTest(path="madvise"):
            mm = _mmap_prefaulted(-1, alloc_bytes, flags)
            try:
                self._assert_resident(mm, alloc_bytes)
            finally:
                mm.close()

        # MAP_POPULATE 分支在 5.14+ 内核上不可达，CI 永远不会跑它；
        # 若不做 force 分支就会带着未测试的代码合入。
        with self.subTest(path="map_populate"), unittest.mock.patch(
            "sglang.srt.mem_cache.storage.mmap.mmap_allocator._has_madv_populate_write",
            return_value=False,
        ):
            mm = _mmap_prefaulted(-1, alloc_bytes, flags)
            try:
                self._assert_resident(mm, alloc_bytes)
            finally:
                mm.close()
```

评论区精华

PR 没有实质 review 评论，仅 xiezhq-hermann 给出 APPROVED（空理由）。评论区只有作者 /rerun-test 与 /rerun-group hicache 的 CI 重跑记录，且重跑全部通过。设计权衡（madvise 优先 + MAP_POPULATE 回退、探针而非 try/except、ENOMEM 传播）均由 PR body 自述，未引发额外讨论。

- 暂无高价值评论线程

风险与影响

- 风险：内核兼容：行为依赖 Linux 5.14+ 的 MADV_POPULATE_WRITE；旧内核探针失败自动回退 MAP_POPULATE，与原行为一致。但若 seccomp 等机制拦截 madvise，探针也会回退，可能只是分配变慢而非失败，风险可控。异常语义收紧：madvise 失败不再被吞，老内核或内存压力场景下原本可能“侥幸继续”的启动会立即抛 OSError，这是有意修复，但属于用户可见行为变化。性能影响：只优化普通 mmap 路径，hugepage 路径未动；探针为一次性单页开销，functools.cache 后每个分配节省一次全量内核遍历。测试局限：新测试只覆盖 64 页小缓冲，未覆盖数百 GB 规模；AMD ROCm 7.2 CI 失败虽与本 PR 无关，但 AMD 平台未被本次变更验证。
- 影响：影响所有使用 alloc_mmap / alloc_shm 的 host pool（不止 HiCache），大内存机器上启动分配时间降低约 13%，且启动路径行为更严格：真实的内存分配失败会立刻暴露，而不是被吞掉后延迟到后续使用阶段。API 未变，调用方无需改动；团队可复用“能力探针 + 回退”和“mock 强制覆盖不可达分支 + mincore 核验不变式”的测试模式。
- 风险标记：启动关键路径，内核版本兼容，异常语义变更，hugepage 路径未覆盖

关联脉络

- 暂无明显关联 PR