PR #34066 完整报告

sgl-project/sglang

perf(unified-memory): batch lazy-compaction mapping lookup

合并时间：2026-08-24 17:14

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34066

执行摘要

- 一句话：批量化映射查找消除 GPU-CPU 同步，unified-memory 吞吐最高提升 128%
- 推荐动作：值得精读。虽然改动只有 55 行，但精准命中了「逐行 .item() 同步」这一典型性能陷阱，_commit_move_batch 中「一次 gather + GPU 异步断言」的组合是可在其它同步敏感路径复用的优化范式；回归测试用 _RejectScalarIndexTensor 代理把「不得逐行读取映射」变成测试期即失败的硬约束，也是防御性测试的好例子。建议结合 #33060 的 Kimi-K3/GB200 数据一起阅读。

功能与动机

PR body 明确指出：Unified-memory lazy compaction 为每个被搬移的幸存者读取 physical_to_virtual[src].item()，而每次标量读取都会同步 GPU 与 CPU，使 compaction 在内存压力下成为主要瓶颈。作者的目标是把「每个幸存者一次同步」压缩为「每批搬移一次批量查询」，从而提升 eviction-heavy 场景下的端到端吞吐。作者还在 Issue 评论中补充：#33060 独立调查了同一同步瓶颈，并提供了 Kimi-K3/GB200 的互补测量。

实现拆解

1. 变更入口与范围：本 PR 只改动 python/sglang/srt/mem_cache/multi_ended_allocator.py 中的 _flush / _commit_move_batch 以及对应单元测试文件 test/registered/unit/mem_cache/test_multi_ended_allocator.py，合计+55/-27，无配置、schema 或部署配套变更。
2. 批量 gather 代替逐行读取：_commit_move_batch 内部使用 v_moveds_t = self.physical_to_virtual[src_pages_t] 一次张量索引取回整批幸存者的 virtual id，替代旧实现中「对每个幸存者 physical_to_virtual[src].item() 后重新组装成张量」的路径；_flush 中同步删除 v_moveds: List[int] 累加器及其 clear() 调用，提交点签名从 (srcs, dsts, v_moveds, latest_event, released_fired) 收紧为 (srcs, dsts, latest_event, released_fired)。
3. 同步消除与断言迁移：原先每个 move batch 至少一次的 .item() 校验被替换为 GPU 侧异步断言 torch._assert_async((v_moveds_t >= 0).all(), "invalid p2v mapping in MultiEndedAllocator._flush")，正常路径不再因校验发生 D2H 同步；负值仍被认定为状态损坏并拒绝静默继续。
4. 测试配套：测试文件新增 _RejectScalarIndexTensor 代理类（整数下标访问即抛 AssertionError）与回归测试 test_lazy_flush_gathers_survivor_mappings_as_one_batch

，验证 flush 只能以批量方式读取映射，并校验搬移后 virtual_to_physical / physical_to_virtual 与 kv.buf 数据一致性。作者另给出全量 allocator 66 passed、eviction/replay parity 80/80 exact match、最大 logprob 差异 0.0 的验证数据。

5. 文档同步：_flush 的 docstring 从「one D2H total」更新为「one free-list D2H plus one mapping D2H per committed move batch」，明确映射 gather 与校验都在 _commit_move_batch 内一次完成。

关键文件：

- python/sglang/srt/mem_cache/multi_ended_allocator.py（模块 分配器；类别 source；类型 core-logic；符号 _flush, _commit_move_batch）：核心改动文件：_flush / _commit_move_batch 将逐幸存者映射读取改为整批 gather，并把有效性校验从 host .item() 迁移到 GPU torch._assert_async，是吞吐提升的直接来源。
- test/registered/unit/mem_cache/test_multi_ended_allocator.py（模块 单元测试；类别 test；类型 test-coverage；符号 _RejectScalarIndexTensor, test_lazy_flush_gathers_survivor_mappings_as_one_batch）：回归测试配套：新增 _RejectScalarIndexTensor 代理与 test_lazy_flush_gathers_survivor_mappings_as_one_batch，把「不得逐行读取映射」固化为测试期硬约束，并验证搬移后映射与 KV 数据一致性。

关键符号：_flush, _commit_move_batch, _RejectScalarIndexTensor, test_lazy_flush_gathers_survivor_mappings_as_one_batch

关键源码片段

python/sglang/srt/mem_cache/multi_ended_allocator.py

核心改动文件：_flush / _commit_move_batch 将逐幸存者映射读取改为整批 gather，并把有效性校验从 host .item() 迁移到 GPU torch._assert_async，是吞吐提升的直接来源。

```
def _commit_move_batch(self, srcs, dsts, latest_event, released_fired):
    """批量提交一次幸存者搬移：一次性 gather 映射、校验并执行 KV 移动。

    旧实现由调用方逐幸存者执行 `physical_to_virtual[src].item()` 并组装
    `v_moveds` 列表；每次 `.item()` 都是一次 GPU 到 CPU 的同步。现在改为
    在本方法内一次张量索引完成全量映射查询。
    """
    with record_function("MultiEndedAlloc._commit_move_batch"):
        # 把 CPU 侧累计的物理页索引一次性搬到 GPU
        src_pages_t = torch.tensor(srcs, dtype=torch.int64, device=self.device)
        dst_pages_t = torch.tensor(dsts, dtype=torch.int64, device=self.device)

        # 关键优化：一次索引操作取回所有幸存者的 virtual id，
        # 替代旧实现中逐行 .item() 再重新组装张量的方式；
        # 标量 .item() 会强制 D2H 同步，是 eviction 场景的主要瓶颈。
        v_moveds_t = self.physical_to_virtual[src_pages_t]

        # 负值意味着被选中的物理页已不是存活幸存者（状态损坏）。
        # 用 GPU 侧异步断言保持不变量，避免为每个 batch 引入 host 同步；
```

```
        # 断言失败会在后续某个同步点统一暴露。
        torch._assert_async(
            (v_moveds_t >= 0).all(),
            "invalid p2v mapping in MultiEndedAllocator._flush",
        )

        # 后续沿用原逻辑：以 v_moveds_t 为 key 更新 virtual_to_physical
        # 反查表，并调用 move_kv_cache 搬移 KV 数据，此处不再展开。

    # _flush 中的提交点：不再维护 v_moveds 累加列表，
    # 映射查询统一推迟到 _commit_move_batch 内批量完成，减少同步次数
    self._commit_move_batch(srcs, dsts, latest_event, released_fired)
    n_moves += len(srcs)
    srcs.clear()
    dsts.clear()
```

test/registered/unit/mem_cache/test_multi_ended_allocator.py

回归测试配套：新增 _RejectScalarIndexTensor 代理与 test_lazy_flush_gathers_survivor_mappings_as_one_batch，把「不得逐行读取映射」固化为测试期硬约束，并验证搬移后映射与 KV 数据一致性。

```
    class _RejectScalarIndexTensor:
        """张量代理：拒绝“一次一行”的映射读取，用于回归测试。"""

        def __init__(self, tensor: torch.Tensor):
            self.tensor = tensor

        def __getattr__(self, name):
            # 透传底层张量的其余属性，让代理可以无缝替换原张量
            return getattr(self.tensor, name)

        def __getitem__(self, index):
            # 整数下标意味着又回到逐行 .item() 的老路，立即失败
            if isinstance(index, int):
                raise AssertionError("physical_to_virtual was read one row at a time")
            return self.tensor[index]

        def __setitem__(self, index, value):
            self.tensor[index] = value


    def test_lazy_flush_gathers_survivor_mappings_as_one_batch(self):
        """Compaction 不能为每个幸存者各同步一次。"""
        _pool, fa, kv = self._make_full(lazy=True)
        values = fa.alloc(12)
        self._stamp_kv(kv, fa, values)
        fa.free(values[1:5].clone())

        # 替换物理到虚拟映射表：一旦出现逐行读取立即抛错
```

```
        physical_to_virtual = fa.physical_to_virtual
        fa.physical_to_virtual = _RejectScalarIndexTensor(physical_to_virtual)
        try:
            self.assertEqual(fa._flush(urgent=True), 4)
        finally:
            fa.physical_to_virtual = physical_to_virtual

        # 校验搬移后映射关系与 KV 数据都保持一致
        for virtual in values.tolist():
            physical = int(fa.virtual_to_physical[virtual].item())
            if physical == -1:
                continue # 已释放
            self.assertEqual(int(fa.physical_to_virtual[physical].item()), virtual)
            self.assertEqual(int(kv.buf[physical].item()), virtual)
```

评论区精华

Review 的核心交锋集中在 _commit_move_batch 的第二版实现上：

> ch-wan（reviewer）：*"Are these .item()s still needed?"*
>
> seokwoosong（作者）：承认第一个 .item() 会在每个 move batch 同步 GPU 与 CPU，正常 compaction 路径上不必要；但有效性校验本身应保留，以避免静默破坏映射——负值代表被选中搬移的页已不是存活幸存者。他提议用 torch._assert_async((v_moveds_t >= 0).all(), ...) 把不变量检查留在 GPU 侧，既消除 host 同步又保留保护。
>
> ch-wan：*"Great! Feel free to ping me on slack when this PR is ready to merge."*

最终作者推送更新并重跑最新上游的 allocator 测试，全部通过、无回归后合并。

- 批量映射查找后 .item() 校验是否还需要 (performance)：保留 GPU 侧异步断言 torch._assert_async，移除 host 侧 .item() 同步校验；ch-wan 认可并准备合并。
- 测试回归确认与合并 (testing)：无回归，PR 合并。

风险与影响

- 风险：
  1. 异步断言的可观测性：torch._assert_async 的失败是异步的，报错可能延迟到后续某个同步点才被抛出，排障路径不如 host 侧断言直观；且依赖 PyTorch 对 _assert_async 的支持与后端行为，若未来换后端需重新验证。
  2. GPU gather 的错误形态变化：v_moveds_t 现在直接来自 GPU 侧张量索引，若 srcs 中出现越界物理索引，错误表现从「断言退出」变为「非法内存访问」，定位难度上升；虽然 _topmost_survivor 已排除 p2v=-1 页，但该假设与 _commit_move_batch 的解耦需要维护者注意。
  3. 核心路径影响面：lazy compaction 位于 MultiEndedAllocator 核心分配路径，影响所有启用 --enable-unified-memory 且发生 L1 eviction 的场景；虽有 66 项单测与 80/80 parity 背书，但 CI 中对高 eviction 压力场景的覆盖仍然有限。 - 影响：对用户与系统：eviction-heavy 的 unified-memory 部署吞吐提升 20.4%~128.2%（RTX 5090 上

Qwen3.5-0.8B / 4B / 9B 实测），内存压力越大收益越明显；静态内存路径完全不受影响。对团队：这是「批量张量索引 + GPU 异步断言」消除同步的示范性优化，与 #33060 的独立测量互相印证同一瓶颈，可为分配器其它存在逐行 .item() 的路径提供参考范式。影响面控制良好：2 个文件、+55/-27，改动集中在 compaction 提交点。 - 风险标记：核心分配路径优化，GPU 异步断言延迟报错，依赖 torch._assert_async 支持

关联脉络

- PR #33060 （评论中提及的独立工作，标题未提供）：本 PR 作者在 Issue 评论中主动提及：#33060 独立调查了同一同步瓶颈，并提供 Kimi-K3/GB200 互补测量，两者共同验证 unified-memory compaction 的同步开销问题。
- PR #36381 Fix SWA ownership across grouped frees: 同属 mem_cache 分配器模块（allocator 子目录）的近期修复，反映该模块在正确性与性能上的并行攻坚，可对照阅读分配器演进脉络。
- PR #36232 Refactor HiCache host pool management: 同为 mem_cache 子系统的结构性演进（主机池管理重构），与本次 compaction 优化共同构成 unified memory 路径的持续优化。