PR #32208 完整报告

sgl-project/sglang

O(1) slot allocation in ReqToTokenPool.alloc()

合并时间：2026-08-12 02:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32208

执行摘要

- 一句话：ReqToTokenPool 槽分配改尾部弹出，热路径由 O(pool) 降为 O(batch)
- 推荐动作：值得精读。核心看点有两个：一是 memory_pool.py 中 alloc() 的尾部弹出与零长度切片防护，二是测试断言如何随分配策略变化而显式放宽。学习点包括 Python 列表切片复杂度、-0 切片陷阱，以及“不透明索引 + req_generation 代际计数解耦”的池设计思路。

功能与动机

alloc() 每调度步准入新请求时执行一次，且运行在单线程调度进程上，直接门控 GPU 分发。原实现 self.free_slots = self.free_slots[need_size:] 会整体复制幸存指针，PR body 指出该行复杂度与剩余空闲槽数成正比而非与实际分配数成正比：“this line is O(len(free_slots)) — proportional to how many free slots remain, not to how many are being allocated (need_size)… a purely wasted allocation-dominated CPU cost on every scheduler iteration”。在 --max-running-requests（4096/16384）高并发稳态下 free_slots 常达数千、每步准入仅个位数到低百位，每次 alloc() 都是纯浪费的 CPU 开销；free() 已是 O(1) 均摊，本 PR 将 alloc 侧补齐。

实现拆解

1. 变更入口与核心替换：python/sglang/srt/mem_cache/memory_pool.py 中 ReqToTokenPool.alloc()，将 self.free_slots[:need_size] 头部切片加整体列表重建，替换为 select_index = self.free_slots[-need_size:] + del self.free_slots[-need_size:]，只触碰被移除的 need_size 个元素，原地删除尾部不触发剩余元素拷贝。
2. 边界防护：need_size == 0 时 free_slots[-0:] 与 free_slots[:] 等价（-0 == 0），会取走整个空闲池；显式 select_index = [] 保证全批次复用槽位时（如 chunked prefill 续批）不消耗任何空闲槽。
3. 测试契约同步：分配顺序从头部改为尾部后，test/registered/unit/mem_cache/test_dllm_fdfo_kv_reuse.py 的两个 extend 测试与 test/registered/unit/hardware_backend/mlx/test_attention_patching.py 的 MLX auxiliary state 测试放宽了原先“新请求固定拿到索引 2/1”的断言，改为只校验复用行索引不变、新行拿到差异化合法槽位，将“分配顺序无语义”显式化为测试契约。
4. 配套调整：无配置、schema、部署变更；贡献者最初提交了独立回归测试与 benchmark 文件，经维护者确认改动足够简洁后移除并迁移至 gist 附录，仓库内回归测试与 CI 均通过。

关键文件：

- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 ReqToTokenPool.alloc）：唯一源码改动点：ReqToTokenPool.alloc() 从头部切片改为尾部弹出，附带 need_size == 0 防护，是本次优化的核心。
- test/registered/unit/mem_cache/test_dllm_fdfo_kv_reuse.py（模块 内存池；类别 test；类型 test-coverage；符号 TestDllmFdfoKvReuse.test_alloc_for_extend_mixed_reuse_allocates_only_fresh_and_writes_rows, TestDllmFdfoKvReuse.test_alloc_for_extend_paged_mixed_reuse_skips_reused_rows）：两个 extend 测试原先断言新请求固定拿到索引 2，分配顺序改变后需放宽为校验复用行不变、新行取到差异化合法槽位。
- test/registered/unit/hardware_backend/mlx/test_attention_patching.py（模块 内存池；类别 test；类型 test-coverage；符号 test_auxiliary_state_req_pool_maps_request_indices）：MLX auxiliary state req pool 的分配顺序断言同样放宽，与主源码改动保持一致。

关键符号：ReqToTokenPool.alloc

关键源码片段

python/sglang/srt/mem_cache/memory_pool.py

唯一源码改动点：ReqToTokenPool.alloc() 从头部切片改为尾部弹出，附带 need_size == 0 防护，是本次优化的核心。

```
def alloc(self, reqs: list[Req]) -> Optional[List[int]]:
    # 收集本批次中已有 req_pool_idx、将复用旧槽位的请求（例如跨 chunk 的
    # chunked prefill 继续请求），它们不需要新分配槽位。
    reusing = [i for i, r in enumerate(reqs) if r.req_pool_idx is not None]
    # 复用请求必须是 chunked 或已有 committed KV；该限制已由 #20476
    # 暂时放宽，故此处不再断言同批次只能有一个 chunked 复用请求。
    assert all(
        reqs[i].inflight_middle_chunks > 0 or reqs[i].kv_committed_len > 0
        for i in reusing
    ), "reusing request must be chunked or have committed KV"

    need_size = len(reqs) - len(reusing)
    if need_size > len(self.free_slots):
        return None
    if need_size > 0:
        # 从尾部弹出 N 个槽位：只搬运被移除的元素，成本 O(need_size)；
        # 从头部弹出并重建列表会整体复制剩余指针，成本 O(len(free_slots))。
        select_index = self.free_slots[-need_size:]
        del self.free_slots[-need_size:]
    else:
        # 单独处理 need_size == 0：free_slots[-0:] 等价于 free_slots[:],
        # 会取到整个列表而不是空列表，所以必须显式返回空列表。
        select_index = []
    offset = 0
    for r in reqs:
        if r.req_pool_idx is None:
```

```
            r.req_pool_idx = select_index[offset]
            self.req_generation[r.req_pool_idx] += 1
            offset += 1
    return [r.req_pool_idx for r in reqs]
```

评论区精华

维护者 xiezhq-hermann 认可改动简洁，建议跳过配套 benchmark 与回归测试文件："good call, the changes seem concise enough and I think we can skip the benchmark and the test file for the changes"，作者随后将测试与 benchmark 移入 gist。Review 中唯一代码级意见是精简内联注释："please shorten the comments as well"（memory_pool.py:310），作者回复"Done."后获得 APPROVED。另外，CI 分区 base-b-test-1-gpu-large 的 test_awq.py（Mixtral-8x7B AWQ INT4）失败被作者定位为 AWQ/Marlin MoE 的 JIT kernel align_single_token.cuh 在 CUDA graph capture 期间的编译错误，与本次内存池记账改动无关。

- 是否保留配套 benchmark 与回归测试文件 (testing): 作者将测试与 benchmark 迁移至 gist 并从 PR 移除，回归测试改为依赖既有单元测试文件。
- alloc() 内联注释冗长需精简 (style): 作者按要求精简注释，随后获得 APPROVED。
- CI 失败（test_awq.py JIT 编译错误）与本 PR 无关 (other): 失败根因在 JIT kernel 编译环境，与本次内存池记账改动无关。

风险与影响

- 风险：
  1. 分配顺序语义变化：尾部弹出与 free() 的尾部追加构成 LIFO 栈，刚释放的槽位最可能被立即复用。PR 明确 req_pool_idx 是不透明行索引、req_generation 已单独跟踪复用，但若存在隐式依赖"编号小的槽位先分配"的监控或外部断言，可能受影响；测试断言放宽是该变化的直接信号。
  2. 负零切片回归风险：need_size == 0 的 guard 一旦在后续重构中丢失，会出现"整池空闲槽被一次性分配"的严重错误，建议保留该分支及注释。
  3. 影响面收敛：free()、available_size()、clear() 与 free_slots 的类型和语义均未变，invariant_checker.py、streaming_session.py 等外部读取方无需修改。
  4. 性能影响：尾弹出为原地删除，无列表整体复制；基准显示池 16384、空闲 14745、单步准入 1-4 时均值由约 32 us 降至约 3 us（约 10.5x），病理场景约 3.9x，小池大批次时新旧实现持平。属调度器 CPU 分发开销优化，端到端吞吐仅在极高并发 / 短 step 场景体现。 - 影响：对用户与系统：不改变模型输出与 KV 缓存内容，端到端正确性无影响；调度器 CPU 账本开销下降，对超大规模部署（--max-running-requests 4096/16384 级别、高并发短 step）的步间开销改善明显，PR body 估算病理场景每百万调度步约节省 28 CPU 秒。对团队与工程：将"分配顺序无语义"上升为测试契约，后续修改分配策略时测试兼容性更好，同时示范了一种低成本消除热路径列表拷贝的模式，可推广到同类空闲池实现。 - 风险标记：调度器热路径变更，槽位分配顺序语义变化，Python 负零切片陷阱防护，测试断言显式放宽

关联脉络

- PR #29792 [HiCache] Fix Mamba track-boundary bookkeeping under overlap scheduling: 同改 python/sglang/srt/mem_cache/memory_pool.py 的 MambaPool 记账逻辑，同属内存池模块的持续硬化。
- PR #34341 [npu] [bugfix] Fix HiCache MHA backup for NPU: 同为 mem_cache 模块（pool_host/mha.py）的修复，说明该模块是调度与缓存一致性的高频改动区。