# PR #28753 完整报告

sgl-project/sglang

Fix/hisparse host backed max request length

合并时间：2026-08-10 16:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28753

## 执行摘要

- 一句话：修复 HiSparse 下长请求被设备池容量误截断
- 推荐动作：值得精读。核心价值在于把分散的容量判断收敛为 ModelRunner.max_token_pool_size 单一出口，并同步修正 capturer 缓冲与 decode 准入两个消费点；测试中「用 object.__new__(ModelRunner) 构造裸实例让 property 描述符正常解析」的夹具技巧也值得借鉴。阅读时建议对照 #34345 了解未覆盖的 max_req_len / init_req_max_new_tokens 路径，形成完整的容量语义地图。

## 功能与动机

HiSparse 将完整序列驻留在 host-backed 逻辑池中，逻辑容量应体现设备池大小乘以 host_to_device_ratio 后的值，而 max_req_input_len 是从 max_total_num_tokens（设备池大小）派生的，导致长上下文请求即使能被 HiSparse 承载也仍被截断在设备容量之下。PR body 与首个 commit 均点明该问题；daii-0818 在 issue 评论中进一步确认：GPU 容量 < request <= HiSparse size_full 时本应接受而被误拒，且该语义在 PD decode 场景仍然必要。

## 实现拆解

本 PR 以「容量语义统一」为主线，分三步落地：

1. 在 ModelRunner 新增统一容量出口（python/sglang/srt/model_executor/model_runner.py）：新增 max_token_pool_size property，当 enable_hisparse 为真且 token_to_kv_pool_allocator 暴露 size_full 时返回该 host-backed 逻辑容量，否则回退到已有的 effective_max_total_num_tokens（SWA 感知）。这样上层调用方无需关心 HiSparse 的扩容细节，所有消费点只认一个逻辑容量。
2. 修正 capturer 缓冲分配：init_routed_experts_capturer 与 init_indexer_capturer 中的 num_tokens 由 self.max_total_num_tokens + self.page_size 改为 self.max_token_pool_size + self.page_size，避免 HiSparse 下请求长度超过设备池容量时 capturer 缓冲越界或断言失败。
3. 修正 PD decode 准入检查（python/sglang/srt/disaggregation/decode.py）：DecodePreallocQueue._check_if_req_exceed_kv_capacity 在 scheduler.enable_hisparse 时改用 scheduler.tp_worker.model_runner.max_token_pool_size 作为容量上限，非 HiSparse 路径仍用 self.max_total_num_tokens，保证行为不回归。

4. 配套测试（test/registered/unit/mem_cache/test_hisparse_max_token_pool_size.py，新增）：注册为 CPU CI 用例，覆盖 max_token_pool_size 的 HiSparse 返回 size_full、size_full 缺失回退、非 HiSparse 委托 effective max、hybrid SWA 优先 full max 四个分支；并用 object.__new__(ModelRunner) 构造裸实例，使 property descriptor 正常解析。decode 侧覆盖 host-backed 容量内准入、超容量拒绝、非 HiSparse 仍按设备池拒绝、rebootstrap 长度（prompt + output）计入容量四种场景。

关键文件：

- python/sglang/srt/model_executor/model_runner.py（模块 模型执行器；类别 source；类型 data-contract；符号 max_token_pool_size, effective_max_total_num_tokens, init_routed_experts_capturer, init_indexer_capturer）：新增 max_token_pool_size property，定义 HiSparse 下 host-backed 逻辑容量语义；同时将路由专家 capturer 与 indexer capturer 的缓冲分配从 max_total_num_tokens 切换到该逻辑容量，是本 PR 的容量契约核心。
- python/sglang/srt/disaggregation/decode.py（模块 解码准入；类别 source；类型 core-logic；符号 _check_if_req_exceed_kv_capacity, _rebootstrap_prefill_len）：DecodePreallocQueue._check_if_req_exceed_kv_capacity 准入容量从设备池 max_total_num_tokens 切换到 HiSparse 逻辑容量，是 PD decode 侧长请求不再被误拒的关键改动。
- test/registered/unit/mem_cache/test_hisparse_max_token_pool_size.py（模块 容量检测；类别 test；类型 test-coverage；符号 _make_model_runner, TestMaxTokenPoolSize, TestCheckIfReqExceedKvCapacity, _make_prealloc_queue）：新增 193 行 CPU 单测，覆盖 max_token_pool_size 的 HiSparse/SWA/fallback 分支以及 decode 准入在 host-backed 容量、超容量拒绝、非 HiSparse 不变、rebootstrap 长度计入四种场景，是本 PR 的核心回归保障。

关键符号：ModelRunner.max_token_pool_size, ModelRunner.effective_max_total_num_tokens, ModelRunner.init_routed_experts_capturer, ModelRunner.init_indexer_capturer, DecodePreallocQueue._check_if_req_exceed_kv_capacity, DecodePreallocQueue._rebootstrap_prefill_len

## 关键源码片段

### python/sglang/srt/model_executor/model_runner.py

新增 max_token_pool_size property，定义 HiSparse 下 host-backed 逻辑容量语义；同时将路由专家 capturer 与 indexer capturer 的缓冲分配从 max_total_num_tokens 切换到该逻辑容量，是本 PR 的容量契约核心。

```
# python/sglang/srt/model_executor/model_runner.py
# HiSparse 容量语义的关键出口：所有上层容量判断都应经此属性取值。
@property
def max_token_pool_size(self):
    """Return the max token pool size considering hybrid swa and hisparse settings."""
    if self.enable_hisparse:
        # HiSparse 把完整序列放在 host-backed 逻辑池中，逻辑容量 = 设备池大小 *
```

```
        # host_to_device_ratio，由 allocator 以 size_full 暴露。若 allocator 尚未
        # 暴露该属性（例如初始化时序导致非 HiSparse allocator 接入），则回退到
        # SWA 感知的 effective_max_total_num_tokens，避免 AttributeError。
        size_full = getattr(self.token_to_kv_pool_allocator, "size_full", None)
        if size_full is not None:
            return size_full
    return self.effective_max_total_num_tokens


# capturer 缓冲分配同步改用逻辑容量，避免 HiSparse 下请求长度超过设备池时
# routed experts / indexer 缓冲越界或断言失败。
def init_routed_experts_capturer(self):
    # ... 省略前置检查 ...
    set_global_experts_capturer(
        RoutedExpertsCapturer.create(
            model=self.model,
            model_config=self.model_config,
            num_tokens=self.max_token_pool_size + self.page_size, # 原为 max_total_num_tokens
            max_running_requests=self.max_running_requests,
            device=self.device,
        )
    )

def init_indexer_capturer(self):
    set_global_indexer_capturer(
        create_indexer_capturer(
            model_config=self.model_config,
            num_tokens=self.max_token_pool_size + self.page_size, # 原为 max_total_num_tokens
            max_running_requests=self.max_running_requests,
            device=self.device,
        )
    )
```

## python/sglang/srt/disaggregation/decode.py

DecodePreallocQueue._check_if_req_exceed_kv_capacity 准入容量从设备池 max_total_num_tokens 切换到 HiSparse 逻辑容量，是 PD decode 侧长请求不再被误拒的关键改动。

```
# python/sglang/srt/disaggregation/decode.py
# PD decode 预分配队列的容量准入检查。
def _check_if_req_exceed_kv_capacity(self, req: Req) -> bool:
    # HiSparse 场景下，请求准入容量取 host-backed 逻辑容量（max_token_pool_size），
    # 而非设备池 max_total_num_tokens。二者在 HiSparse 下可能相差 host_to_device_ratio 倍，
    # 若仍按设备容量校验，会把可被 HiSparse 承载的长上下文请求误判为超限并 abort。
    if self.scheduler.enable_hisparse:
        capacity = self.scheduler.tp_worker.model_runner.max_token_pool_size
    else:
        # 非 HiSparse 路径保持原行为，避免本改动影响常规 decode 准入。
        capacity = self.max_total_num_tokens
```

```
input_len = self._rebootstrap_prefill_len(req)
if input_len > capacity:
    message = f"Request {req.rid} exceeds the maximum number of tokens: {input_len} >
    {capacity}"
    logger.error(message)
    prepare_abort(req, message, status_code=HTTPStatus.BAD_REQUEST)
    self.scheduler.output_streamer.stream_output([req], req.return_logprob)
    return True
# ... 后续 SWA tail 预分配检查与原有逻辑保持一致 ...
return False
```

## 评论区精华

1. gemini-code-assist[bot]（high）：指出 max_token_pool_size 变大后，init_routed_experts_capturer 与 init_indexer_capturer 仍按 max_total_num_tokens + page_size 分配缓冲，HiSparse 下会越界或断言失败——已在后续 commit 中改为 max_token_pool_size + page_size。
2. gemini-code-assist[bot]（medium）：建议 decode.py 直接使用 self.scheduler.enable_hisparse 而非 getattr(model_runner, "enable_hisparse", False)，与文件内其他 HiSparse 判断保持一致——最终代码采用了该建议。
3. huangtingwei9988：第一轮要求补单元测试；随后指出 test_non_hisparse_hybrid_swa_prefers_full_max 用 SimpleNamespace 调用 ModelRunner.max_token_pool_size 会因属性描述符不解析而抛 AttributeError，建议改用 object.__new__(ModelRunner) 让两个 property 都走正常描述符查找——作者据此重构了测试夹具。
4. daii-0818：验证发现下游 TpModelWorker.max_req_len 与 Scheduler.init_req_max_new_tokens() 仍使用 GPU 驻留容量，属于本 PR 未覆盖的遗漏点，并已开跟进 PR #34345 附带 CPU 回归测试。

- capturer 缓冲分配越界风险 (correctness)：已修复：两处 num_tokens 均改用 self.max_token_pool_size + self.page_size。
- 使用 scheduler.enable_hisparse 保持一致性 (style)：已采纳，最终代码使用 self.scheduler.enable_hisparse 分支。
- 补单元测试 (testing)：已补充 test_hisparse_max_token_pool_size.py 并注册 CPU CI。
- SimpleNamespace 绕过 property 描述符导致 AttributeError (testing)：已修复：_make_model_runner 改用 object.new(ModelRunner) 并 setattr 注入属性，使两个 property 都走正常描述符查找。
- 下游仍有两处使用 GPU 驻留容量 (correctness)：本 PR 未覆盖，交由 #34345 跟进修复并附带 CPU 回归测试。

## 风险与影响

- 风险：
  1. 下游消费点未全覆盖：daii-0818 指出 TpModelWorker.max_req_len 与 Scheduler.init_req_max_new_tokens() 仍按 GPU 驻留容量计算，HiSparse 下长请求

可能在其他入口被截断，需由 #34345 收尾。

2. capturer 缓冲内存上升：init_routed_experts_capturer / init_indexer_capturer 的 num_tokens 改大后，routed expert 捕获与 indexer 捕获的缓冲将按 host-backed 逻辑容量分配，在设备内存有限时可能增加显存压力。
3. 契约依赖：max_token_pool_size 依赖 allocator.size_full 是否存在及 enable_hisparse 标志是否与 allocator 实际类型一致；若初始化时序导致 HiSparse allocator 尚未接线，会静默回退到设备容量，行为正确但语义不达预期。
4. 回归面：decode 准入从直接读 max_total_num_tokens 改为经 scheduler.tp_worker.model_runner 间接取容量，引入跨对象依赖，若 tp_worker 或 model_runner 未就绪会抛属性错误（测试已用 mock 覆盖主路径）。 - 影响：对 HiSparse 用户是功能性修复：长上下文请求不再被设备池上限误拒，可用请求长度提升 host_to_device_ratio 倍，直接利好超长序列离线批处理与 PD 解耦部署。对非 HiSparse 用户无行为变化，但 max_token_pool_size 成为新的统一容量出口，后续容量相关逻辑都应优先经它取值。对团队而言，本 PR 确立了「逻辑容量 vs 物理设备容量」分离的建模方式，并暴露了两处下游遗漏点（#34345），提示容量语义类改动需要全局检索消费点。 - 风险标记：下游消费点未全覆盖（#34345），capturer 缓冲内存上升，容量契约依赖 allocator 时序，新增跨对象依赖

# 关联脉络

- PR #34345 Follow-up: use GPU-resident capacity in TpModelWorker.max_req_len and Scheduler.init_req_max_new_tokens (comment-referenced): daii-0818 在 issue 评论中明确表示该 PR 是本 PR 的跟进，修复下游仍使用 GPU 驻留容量的两处遗漏点，并附带 CPU 回归测试。
- PR #33484 perf(hisparse): fuse the DSv4 value and scale swap-in copy on ROCm: 同为 HiSparse 功能线，聚焦 swap-in 数据通路性能，与本 PR 的逻辑容量语义互补，共同完善 HiSparse 长序列支持。
- PR #33085 perf(hisparse): 128-bit non-temporal swap-in copy on ROCm: HiSparse 数据通路优化的早期 PR，说明 HiSparse 主从交换路径持续演进，本 PR 补齐其容量准入语义。