# PR #23862 完整报告

sgl-project/sglang

Fix --mem-fraction-static not accounting for EAGLE draft model KV cache

合并时间：2026-06-13 01:35

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23862

## 执行摘要

- 一句话：修复 EAGLE 下静态显存分数未计入 draft KV 缓存导致 OOM
- 推荐动作：此 PR 是一次重要的内存管理重构，修复了长期存在的 draft 模型显存预算缺失问题。设计上将初始化阶段清晰分离，提高了可维护性和可预测性。建议所有使用推测解码的团队精读，尤其是 init_model_worker 的编排逻辑和各 worker 的 alloc_memory_pool/init_backends 实现。

## 功能与动机

根据 PR 描述，当 EAGLE 推测解码启用时，--mem-fraction-static 仅控制目标模型的 KV 缓存池。draft 模型的权重和 KV 缓存未计入预算，导致目标 KV 池过分配（因内存分析在 draft 权重加载前进行），且 draft KV 池在同一 max_total_num_tokens 下额外分配。之前通过粗糙的 4GB/6GB 启发式预留缓解，但对于大型模型或用户指定的 --mem-fraction-static 仍然不足。

## 实现拆解

1. 提前获取 draft 模型层数（model_runner.py）：在 __init__ 中增加 eagle_draft_num_layers 属性，通过 _build_model_config 在权重加载前读取 draft 配置，记录层数用于后续 KV 池容量核算。同时保存 pre_model_load_memory 快照。
2. 调度器初始化三阶段化（scheduler.py）：新增 init_target_memory_pool、init_memory_pools、init_all_backends 三个方法，在 init_model_worker 中按序调用：先加载目标权重（init_tp_model_worker），再加载 draft 权重（maybe_init_draft_worker），然后分配 KV 池（此时内存分析已见所有权重），最后初始化后端和 CUDA 图。
3. draft worker 解耦池分配与后端初始化（多个 speculative worker）：Eagle、multi-layer Eagle、DFlash、FrozenKVMTP 等 worker 的构造器仅加载权重，不再直接分配池或初始化后端；新抽取的 alloc_memory_pool 和 init_backends 方法由调度器在适当时机调用，接收目标池的配置。
4. tp_worker 通用化池分配（tp_worker.py）：新增 alloc_memory_pool 和 init_backends 方法，支持外部传入池对象或创建新池，并统一对 model_runner_list 中所有 runner 执行操作。get_worker_info 后移 max_req_len 计算到池分配之后。
5. 移除启发式预留（server_args.py）：删除 speculative_draft_load_format 逻辑中针对 EAGLE/STANDALONE 的 4GB/6GB 自动 mem_fraction_static 覆盖，因为新机制已精确

核算 draft 模型开销。

6. 配套测试调整：新增 test_model_config_shapes.py 验证 ModelConfig._derive_model_shapes 中的 head_dim 派生逻辑；更新 test_eagle_worker_v2_topk1_fastpath.py 适配新的后端初始化接口。

关键文件：

- python/sglang/srt/model_executor/model_runner.py（模块 模型执行器；类别 source；类型 core-logic；符号 _build_model_config, initialize, alloc_memory_pool, init_backends）：核心变更文件：在 __init__ 中提前加载 draft 模型层数用于 KV 池核算，保存 pre_model_load_memory 供内存分析使用；新增 eagle_draft_num_layers 属性。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 init_target_memory_pool, init_memory_pools, init_all_backends）：调度器新增三个方法编排初始化阶段，是流程重构的核心调度节点。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 Eagle 工作器；类别 source；类型 core-logic；符号 alloc_memory_pool, init_backends, spec_v2_attn_backends）：Eagle draft worker 重构：构造仅加载权重，池分配和后端初始化延后到新方法中。
- python/sglang/srt/managers/tp_worker.py（模块 TP 工作器；类别 source；类型 core-logic；符号 alloc_memory_pool, init_backends）：TP worker 新增 alloc_memory_pool 和 init_backends 通用方法，支持分离调用和外部池注入。
- test/registered/unit/configs/test_model_config_shapes.py（模块 配置测试；类别 test；类型 test-coverage；符号 _make_text_config, TestModelConfigShapes, _derive_shapes, test_optional_head_dims_default_when_none）：新增单元测试验证 ModelConfig 形状推导逻辑，确保 head_dim 等派生正确。

关键符号：_build_model_config, initialize, alloc_memory_pool, init_backends, init_target_memory_pool, init_memory_pools, init_all_backends, spec_v2_attn_backends, _derive_model_shapes, _make_text_config, _derive_shapes, test_optional_head_dims_default_when_none, test_explicit_head_dims_are_preserved

## 关键源码片段

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

核心变更文件：在 __init__ 中提前加载 draft 模型层数用于 KV 池核算，保存 pre_model_load_memory 供内存分析使用；新增 eagle_draft_num_layers 属性。

```
# python/sglang/srt/model_executor/model_runner.py
# 在 __init__ 中新增：提前获取 draft 模型层数
self.eagle_draft_num_layers = None # 新增属性，记录 draft 模型层数
if (
    (self.spec_algorithm.is_eagle() or self.spec_algorithm.is_standalone())
    and not self.is_draft_worker
    and server_args.speculative_draft_model_path
):
    # 在权重加载前读取 draft 配置，获取层数用于 KV 缓存大小核算
    draft_model_config = self._build_model_config(
```

```
            server_args,
            model_path=server_args.speculative_draft_model_path,
            model_revision=server_args.speculative_draft_model_revision,
            is_draft_model=True,
        )
        num_nextn_predict_layers = draft_model_config.num_nextn_predict_layers
        if num_nextn_predict_layers is not None:
            self.eagle_draft_num_layers = int(num_nextn_predict_layers)
        else:
            # fallback: 取 hidden_layers 和 attention_layers 的较大值
            self.eagle_draft_num_layers = int(
                max(
                    draft_model_config.num_hidden_layers,
                    draft_model_config.num_attention_layers,
                )
            )

    # 保存权重加载前的可用内存快照，供 alloc_memory_pool 使用
    self.pre_model_load_memory = self.init_torch_distributed()
```

## python/sglang/srt/managers/scheduler.py

调度器新增三个方法编排初始化阶段，是流程重构的核心调度节点。

```
    # python/sglang/srt/managers/scheduler.py
    # 新增的三阶段初始化方法:
    def init_target_memory_pool(self):
        """仅分配目标KV池（如尚未分配）"""
        if (
            self.tp_worker.model_runner.memory_pool_config is not None
            and self.tp_worker.model_runner.req_to_token_pool is not None
            and self.tp_worker.model_runner.token_to_kv_pool_allocator is not None
        ):
            return
        self.tp_worker.alloc_memory_pool()

    def init_memory_pools(self):
        """分配所有worker的KV池（目标和draft）"""
        self.init_target_memory_pool()
        if self.draft_worker is not None:
            pool, allocator = self.tp_worker.get_memory_pool()
            self.draft_worker.alloc_memory_pool(
                memory_pool_config=self.tp_worker.model_runner.memory_pool_config,
                req_to_token_pool=pool,
                token_to_kv_pool_allocator=allocator,
            )

    def init_all_backends(self):
        """初始化注意力后端和CUDA图"""
        self.tp_worker.init_backends()
```

```
        if self.draft_worker is not None:
            self.draft_worker.init_backends()

    def init_model_worker(self):
        # 1. 加载目标权重
        self.init_tp_model_worker()
        if self.spec_algorithm.is_frozen_kv_mtp():
            # Frozen-KV MTP 需要目标 KV 池提前就绪
            self.init_target_memory_pool()
        # 2. 加载 draft 权重
        self.maybe_init_draft_worker()
        # 3. 分配所有 KV 池（此时所有权重已加载，内存分析准确）
        self.init_memory_pools()
        # 4. 初始化后端和 CUDA 图
        self.init_all_backends()
        # ... 后续逻辑不变 ...
```

## python/sglang/srt/speculative/eagle_worker_v2.py

Eagle draft worker 重构：构造仅加载权重，池分配和后端初始化延后到新方法中。

```
    # python/sglang/srt/speculative/eagle_worker_v2.py
    # 构造中仅加载 draft 模型权重，不再分配 KV 池和初始化后端
    # 原 __init__ 中与池分配、后端初始化相关的代码被移除
    class EagleDraftWorker:
        def __init__(self, ..., target_worker):
            # ... 前面复制参数 ...
            # 只加载 draft 模型权重（不传 pool 和 memory_pool_config）
            self.draft_worker = TpModelWorker(
                server_args=server_args,
                gpu_id=gpu_id,
                tp_rank=tp_rank,
                is_draft_worker=True,
                # 不再传入 req_to_token_pool、token_to_kv_pool_allocator、memory_pool_config
                ...
            )
            self.draft_runner = self.draft_worker.model_runner
            # ... 其他一些配置 ...

        # 新增：由调度器在适当时候调用，分配 KV 池
        def alloc_memory_pool(self, memory_pool_config=None, req_to_token_pool=None, token_to_
        kv_pool_allocator=None):
            self.req_to_token_pool = req_to_token_pool
            self.token_to_kv_pool_allocator = token_to_kv_pool_allocator
            self.draft_worker.alloc_memory_pool(
                memory_pool_config=memory_pool_config,
                req_to_token_pool=req_to_token_pool,
                token_to_kv_pool_allocator=token_to_kv_pool_allocator,
            )
            self.init_token_map()
```

```
        self.init_lm_head()

    # 新增：由调度器在池分配后调用，初始化后端和 CUDA 图
    def init_backends(self):
        with self.draft_tp_context(self.draft_runner.tp_group), ...:
            self.draft_worker.init_backends(disable_cuda_graph=True)
            self.init_attention_backend()
            if check_cuda_graph_backend(...):
                self.draft_runner.init_prefill_cuda_graph(force_for_draft_worker=True)
            self.init_cuda_graphs()
        if (c := self.draft_runner.canary_manager) is not None:
            c.mark_init_finished()
```

## 评论区精华

Review 中 merrymercy 提出了三个关键评论：

- 在 model_config.py 中，询问将 head_dim 写回到 hf_text_config 是否有必要，cctry 确认不需要并承诺删除。
- 在 model_runner_kv_cache_mixin.py 中，指出注释将 draft 模型权重描述为“动态部分”不够准确，cctry 解释这是向后兼容的临时措施，若计入 draft 则静态分数可能超过 0.9 导致测试失败，故保留注释说明。
- 在 frozen_kv_mtp_worker_v2.py 中，建议将 init_backends 拆分为更具体命名的小函数，cctry 同意并拆分。
- _derive_model_shapes 中写回 head_dim 到 hf_text_config 的必要性 (design): 将移除写回操作，保持派生逻辑仅存储在 self.head_dim 上。
- _profile_available_bytes 注释描述 draft 权重为动态部分是否准确 (documentation): 保留注释作为临时兼容措施的说明，不修改行为。
- init_backends 命名模糊，建议拆分 (design): 将 init_backends 拆分为多个职责单一的小函数。

## 风险与影响

- 风险：
  1. 初始化顺序依赖：三阶段化依赖于调度器正确调用顺序，若后续新增 worker 类型未按此模式实现，可能引发未初始化池访问。
  2. 移除启发式预留：此前依赖 4GB/6GB 自动预留的配置（即使不合理）可能因移除而 OOM，需要用户重新调整 --mem-fraction-static。
  3. 池共享契约变更：draft worker 构造不再接收 req_to_token_pool 和 token_to_kv_pool_allocator，而是通过 alloc_memory_pool 传递，若第三方扩展未更新将出错。
  4. 测试覆盖风险：虽然新增了单元测试，但多 worker 交互场景（如 EAGLE+DFLASH 组合）可能覆盖不足，引入回归。

5. 启动时间增加：阶段化可能导致额外同步开销，但影响通常很小。 - 影响：影响范围：所有使用 EAGLE 或其他推测解码（Standalone/DFLASH）的用户。启动时 OOM 问题消除，显存利用更精确。影响程度：较大。涉及核心初始化流程重构，但对外部 API 无破坏性变更；用户无需修改配置即可受益。团队影响：开发者需要理解新的三阶段初始化模式，future speculative worker 应遵循此模式。

- 风险标记：核心路径变更，移除启发式配置，兼容性风险，测试覆盖不足

## 关联脉络

- 暂无明显关联 PR