PR #35944 完整报告

sgl-project/sglang

Pin scheduler metadata before asynchronous H2D copies

合并时间：2026-08-27 13:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35944

执行摘要

- 一句话：调度器元数据改用固定内存，解锁异步 H2D 拷贝
- 推荐动作：值得精读。该 PR 是调度器重叠优化的基础性改动，展示了 pin_memory 与 non_blocking 搭配使用的关键细节：只有源数据本身在固定内存中，non_blocking H2D 才能避免 CUDA 内部同步 staging。建议同时关注 #34515 的完整实现，并留意 _make_num_token_non_padded helper 的抽取方式——它统一了多个调用点的构造逻辑，是后续扩展 pin 内存覆盖面的可借鉴模式。

功能与动机

PR body 明确指出："Passing pageable host tensors to .to(device, non_blocking=True) still requires CUDA to stage the source through pinned memory. That staging can block the scheduler thread and reduce CPU/GPU overlap." 因此将调度器产生的 metadata 直接放在固定内存中，使 H2D 拷贝能按预期异步入队，属于调度器与 GPU 执行重叠优化的前置工作，源自 #34515 中实现并验证过的方案。

实现拆解

变更入口为调度热路径上的 4 处 host 元数据构造点，统一在 torch.tensor 上增加 pin_memory=is_pin_memory_available(device)，随后保留 .to(device, non_blocking=True)，确保 CUDA 不需要再同步暂存页式源。

1. *extend* 分配路径：python/sglang/srt/mem_cache/allocation.py 的 alloc_for_extend() 中，prefix_lens_cpu、extend_lens_cpu 以及 req_pool_indices_cpu 三个张量由普通页式构造改为固定内存构造，覆盖调度器每轮 *extend* 最常执行的 H2D 元数据拷贝。
2. forward-batch 元数据：python/sglang/srt/model_executor/forward_batch_info.py 的 ForwardBatch.init_new() 为 num_token_non_padded、*extend_seq_lens*、*extend_prefix_lens* 增加 pin_memory；init_mlp_sync_metadata() 中 global_num_tokens_gpu、global_num_tokens_for_logprob_gpu 也同步处理，让 DP-attention/MLP 同步路径受益。
3. DSpark draft 侧：python/sglang/srt/speculative/dspark_components/dspark_draft.py 抽取 _make_num_token_non_padded() helper，统一构造 num_token_non_padded，并在 _fill_dp_moe_sync_metadata() 中为 global_num_tokens_gpu / global_num_tokens_for_logprob_gpu 加 pin_memory；同时修复了原逻辑在

enable_num_token_non_padded() 关闭时仍会在 device 上构造张量的分支问题。

4. DSpark prefill 侧：python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py 的 _forward_prefill() 中 ctx_lens、draft_seq_lens 改为固定内存构造后再异步拷贝，服务于 unified_kv 注入前的 H2D 路径。
5. fallback 与验证：所有入口都通过 is_pin_memory_available(device) 判断，CPU、NPU 等不支持固定内存的平台自动回退到普通分配，行为不变。本 PR 未新增直接单元测试，验证依赖 #34515 的 E2E 测试与独立负载测试（Qwen3.5 AgentX 结果在 3% 波动阈值内）。

关键文件：

- python/sglang/srt/speculative/dspark_components/dspark_draft.py（模块 投机解码；类别 source；类型 core-logic；符号 _make_num_token_non_padded, _fill_dp_moe_sync_metadata）：DSpark draft 侧元数据（num_token_non_padded、DP/MoE 同步 token 数）全部改为固定内存构造，并抽取 _make_num_token_non_padded helper 统一逻辑，是 review 中唯一讨论点所在文件。
- python/sglang/srt/model_executor/forward_batch_info.py（模块 批次元数据；类别 source；类型 data-contract；符号 ForwardBatch.init_mlp_sync_metadata, ForwardBatch.init_new）：forward-batch 核心数据契约文件，init_new 与 init_mlp_sync_metadata 覆盖每 forward 必走的元数据构造路径，影响面最广。
- python/sglang/srt/speculative/dspark_components/dspark_worker_v2.py（模块 投机解码；类别 source；类型 core-logic；符号 DSparkWorkerV2._forward_prefill）：DSpark prefill 的 ctx_lens/draft_seq_lens 由直接 device 构造改为固定内存构造，服务于 unified_kv 注入路径，是投机解码 prefill 的重要 H2D 位置。
- python/sglang/srt/mem_cache/allocation.py（模块 内存分配；类别 source；类型 core-logic；符号 alloc_for_extend）：extend 分配路径的核心入口，alloc_for_extend 每轮 extend 都会执行，prefix/extend lens 与 req_pool_indices 的固定内存构造直接影响调度器 CPU/GPU 重叠。

关键符号：_make_num_token_non_padded, _fill_dp_moe_sync_metadata, ForwardBatch.init_mlp_sync_metadata, ForwardBatch.init_new, alloc_for_extend, DSparkWorkerV2._forward_prefill

关键源码片段

python/sglang/srt/speculative/dspark_components/dspark_draft.py

DSpark draft 侧元数据（num_token_non_padded、DP/MoE 同步 token 数）全部改为固定内存构造，并抽取 _make_num_token_non_padded helper 统一逻辑，是 review 中唯一讨论点所在文件。

```
def_make_num_token_non_padded(     num_tokens: int, device: str | torch.device ) -> Optional[torch.Tensor]:     # 统一构造 num_token_non_padded 元数据的 helper。     # 未开启该特性时返回 None，调用方自行跳过赋值，     # 避免在关闭场景下仍在 device 上构造无用张量。     if not enable_num_token_non_padded():         return None     # 关键点：non_blocking H2D 拷贝若源是 pageable 内存，     # CUDA 仍会同步 staging 到固定内存，阻塞调度线程。     # 直接 pin_memory 构造可让后续 .to(device, non_blocking=True)     #
```

真正异步入队，提升 CPU/GPU 重叠度。 return torch.tensor(num_tokens, dtype=torch.int32, pin_memory=is_pin_memory_available(device),).to(device, non_blocking=True) def_fill_dp_moe_sync_metadata(self, forward_batch: ForwardBatch, batch: ScheduleBatch) -> None: # 先设置 graph 资格，DP-MoE 专用元数据随后填充。 forward_batch.can_run_dp_cuda_graph = batch.can_run_dp_cuda_graph if not self._dp_moe_sync or batch.global_num_tokens is None: return gnt, gnt_logprob = spec_scale_global_num_tokens(self._draft_block_spec_info, batch.global_num_tokens, batch.global_num_tokens_for_logprob,) num_tokens = forward_batch.input_ids.numel() # 复用统一的 helper，避免此处重复实现 pin_memory 判断。 num_token_non_padded = _make_num_token_non_padded(num_tokens, device) if num_token_non_padded is not None: forward_batch.num_token_non_padded = num_token_non_padded forward_batch.num_token_non_padded_cpu = num_tokens # DP/MoE 同步用的全局 token 数同样需要固定内存， # 否则 non_blocking H2D 会被 CUDA 内部 staging 拖慢。 pin_memory = is_pin_memory_available(device) forward_batch.global_num_tokens_gpu = torch.tensor(gnt, dtype=torch.int64, pin_memory=pin_memory).to(device, non_blocking=True) forward_batch.global_num_tokens_for_logprob_gpu = torch.tensor(gnt_logprob, dtype=torch.int64, pin_memory=pin_memory).to(device, non_blocking=True)

python/sglang/srt/model_executor/forward_batch_info.py

forward-batch 核心数据契约文件，init_new 与 init_mlp_sync_metadata 覆盖每 forward 必走的元数据构造路径，影响面最广。

```
def init_mlp_sync_metadata(
    self, batch: ScheduleBatch, device: Union[str, torch.device]
) -> None:
    """Populate per-rank token counts for DP-attention MLP synchronization."""
    if batch.global_num_tokens is None:
        return
    # 投机解码场景下需按 spec 系数换算 token 数。
    if self.spec_info is not None:
        global_num_tokens, global_num_tokens_for_logprob = (
            spec_scale_global_num_tokens(
                self.spec_info,
                batch.global_num_tokens,
                batch.global_num_tokens_for_logprob,
            )
        )
    else:
        global_num_tokens = batch.global_num_tokens
        global_num_tokens_for_logprob = batch.global_num_tokens_for_logprob

    self.original_global_num_tokens_cpu = batch.global_num_tokens
    self.global_num_tokens_cpu = global_num_tokens
```

```
# 这两个 token 计数在每个 forward 都被拷贝到 GPU,
# 用于 DP-attention 的 MLP 同步；固定内存避免同步 staging。
pin_memory = is_pin_memory_available(device)
self.global_num_tokens_gpu = torch.tensor(
    global_num_tokens, dtype=torch.int64, pin_memory=pin_memory
).to(device, non_blocking=True)
self.global_num_tokens_for_logprob_cpu = global_num_tokens_for_logprob
self.global_num_tokens_for_logprob_gpu = torch.tensor(
    global_num_tokens_for_logprob,
    dtype=torch.int64,
    pin_memory=pin_memory,
).to(device, non_blocking=True)
self.can_run_dp_cuda_graph = batch.can_run_dp_cuda_graph
```

评论区精华

Review 仅有一条实质性讨论：kpham-sgl 在 dspark_draft.py 上指出 "Nit: I think there is also a h2d here"（还有一处 H2D 拷贝未覆盖），作者 weireweire 回复 "Thanks! fixed." 并已修复合入，该线程已解决。此外 YAMY1234 的验证结论表明 Qwen3.5 AgentX 负载 Total TPS/GPU 为 117,090.28 vs 117,919.87 基线（-0.704%），P90 TPS/User 为 107.111 vs 108.131（-0.943%），均在 3% 阈值内，判定无性能回归。两位 reviewer（kpham-sgl、hnyls2002）均批准。

- dspark_draft.py 中遗漏的 H2D 拷贝 (style): 作者 weireweire 回复 "Thanks! fixed." 并已修复合入。

风险与影响

- 风险：
  1. 热路径分配开销：固定内存分配本身比普通分配略重，但本 PR 涉及的元数据都是标量或小向量（每 batch 数十到数百元素），开销可忽略，收益大于成本；且 is_pin_memory_available 在 CPU/ 不支持平台自动回退。
  2. 张量生命周期：torch.tensor(..., pin_memory=True) 后链式 .to(device, non_blocking=True)，只要拷贝已在流中排队，源张量释放不受影响（CUDA 会保持源直到拷贝完成）；各处调用方式一致，风险低。
  3. 缺少直接单元测试：本 PR 没有配套单测，回归主要依赖 #34515 的 E2E 与独立负载验证，CI 出现过失败（Run #32802810529 等），经重跑后通过。
  4. 覆盖范围：review 中 kpham-sgl 指出过遗漏的 H2D 点，说明同模式可能还有其他未覆盖位置（如 forward_batch_info.py 中 idle 分支等），后续需留意同类拷贝。 - 影响：影响范围集中在调度器热路径：alloc_for_extend（每轮 extend 必走）、ForwardBatch.init_new（每 forward 必走）、DP-MoE 同步元数据、DSpark 投机解码的 draft/prefill 阶段。对用户侧，配合 #34515 的完整调度重叠方案可带来显著 E2E 吞吐提升（完整方案 40%），单独合入也不引入可测回归；对团队侧，确立了一种可复用的优化模式——在调度线程构造 host 元数据时直接使用固定内存，为后续同类 H2D 路径优化提供了范本，并推动调度器与 GPU 执行的重叠演进。 - 风险标记：调度热路径变

更，缺少直接单元测试，依赖 E2E 集成验证

关联脉络

- PR #34515 Broad scheduler-overlap work: PR body 明确说明本改动是从 #34515 提取的聚焦变更，该工作 E2E 吞吐提升 40%，本 PR 的验证依赖其 E2E 测试。
- PR #35947 Publish gated DSV4 DFLASH-family target-prefill read completion: 同为调度器与 GPU 执行重叠方向的工作，涉及 prefill 完成信号提前发布与调度重叠，与本 PR 的 H2D 异步化目标一致。