PR #30177 完整报告

sgl-project/sglang

[Feature] Support return_hidden_states="last"

合并时间：2026-08-02 15:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30177

执行摘要

- 一句话：新增 return_hidden_states="last"，仅返回末 token 向量以降低载荷
- 推荐动作：值得精读，是 hidden-states 服务化能力的关键演进。重点关注三处设计决策：(1) 固定 server 级 capture ceiling 取代按需重捕获，避免图重建抖动；(2) radix cache 语义对 prefill 切片偏移步长的约束（extend_input_len_per_req 而非 len(origin_input_ids)）；(3) EAGLE tc_piecewise 与 FP4/TRTLLM-MoE 死图前科（#28870）对 skip guard 的要求。测试文件 test_batch_result_processor_hidden_states.py 与 test_hidden_state_graph_recapture.py 是理解边界条件的最佳入口。

功能与动机

PR body 明确说明 SGLang 正被用作表示抽取层（二级分类器、安全头、reranker、校准模型、聚类、检索、监控），这类负载只需要最终 token 的 hidden state 作为紧凑语义特征；而现有 boolean 接口只能二选一：不返回或返回全序列，后者在 payload、序列化、网络与存储上明显浪费，且要求客户端自行切片。PR 提出 opt-in 的 "last" 模式，在保留旧行为的同时让调用方直接拿到单向量。

实现拆解

1. 请求级模式规范化：python/sglang/srt/managers/io_struct.py 新增 _normalize_return_hidden_states，校验 per-prompt 模式列表长度与取值、按 parallel_sample_num 展开并支持标量广播；schedule_batch.py 定义 ReturnHiddenStatesMode = Union[bool, Literal["last"]]，新增 get_return_hidden_states_mode / get_request_return_hidden_states_mode / get_batch_return_hidden_states_mode / need_return_hidden_states 4 个函数，把请求值映射为 CaptureHiddenMode.NULL / FULL / LAST，Req 构造时缓存规范化模式，ScheduleBatch 的 init_new / filter_batch / merge_batch / copy 全部改为用 max 聚合批级模式并派生旧布尔字段。
2. 服务端配置与请求校验：server_args.py 新增 --return-hidden-states-mode（last / full），--enable-return-hidden-states 兼容映射为 full；_handle_return_hidden_states_mode 显式拒绝 {None, "last", "full"} 之外的值；tokenizer_manager.py 按 server 上限校验请求模式，超限直接拒绝。
3. 图捕获策略重构：forward_batch_info.py 新增 get_server_return_hidden_states_mode 与 get_required_capture_hidden_mode（叠加 spec_info 的捕获需求）；decode /

prefill / CPU 三个图 runner 启动时把 capture_hidden_mode 固定为 server 配置的最大模式，旧的 recapture_if_needed 替换为 _validate_capture_hidden_mode——超限抛 RuntimeError，弱模式请求复用已有图，不再触发重捕获。prefill runner 的 EAGLEtc_piecewise skip guard 改为依赖有效 capture mode。

4. 结果处理与响应组装：batch_result_processor.py 的 prefill 切片按 extend_input_len_per_req[i] 逐行前进（对 radix cache 命中与 inflight middle chunk 均正确），decode 新增 _append_decode_hidden_states 使 "last" 模式覆盖写入单向量并用 finished_len 排除 spec 越界；output_streamer.py 与 OpenAI 侧按模式返回一维向量（"last"）或完整序列（True）。
5. 测试与示例配套：新增 test_hidden_state_graph_recapture.py（server ceiling、强弱模式复用、超限拒绝、spec override）、test_batch_result_processor_hidden_states.py（middle chunk 偏移、last 有界存储），扩展 test_io_struct.py、test_hidden_state_server_mode.py、核心 test_hidden_states.py（混合模式、暖 cache 数值一致性），并更新 examples/runtime/hidden_states/ 两个示例。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 批调度；类别 source；类型 dependency-wiring；符号 get_return_hidden_states_mode, get_request_return_hidden_states_mode, get_batch_return_hidden_states_mode, need_return_hidden_states）：定义 ReturnHiddenStatesMode 类型与 4 个模式映射函数，Req 与 ScheduleBatch 生命周期各环节全部改为按 CaptureHiddenMode 聚合，是本 PR 的数据契约核心。
- python/sglang/srt/managers/scheduler_components/batch_result_processor.py（模块 结果处理；类别 source；类型 core-logic；符号 _append_decode_hidden_states, _get_prefill_hidden_capture_mode）：prefill 切片偏移与 decode 有界存储的实现地，集中体现了 radix cache 语义、inflight middle chunk 和 spec 越界三个易错点，是 review 讨论最密集的文件。
- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 图执行器；类别 source；类型 data-contract；符号 recapture_if_needed, _validate_capture_hidden_mode）：图捕获策略重构的核心载体：删除 recapture_if_needed 的动态重捕获，改为启动时固定 server 级 capture ceiling 的静态校验，消除请求模式交替导致的图重建抖动。
- python/sglang/srt/model_executor/cpu_graph_runner.py（模块 图执行器；类别 source；类型 data-contract；符号 recapture_if_needed, _validate_capture_hidden_mode）：与 decode runner 同步的图策略重构；额外处理 draft worker 的 NULL 强制与 CPU 图捕获时的 spec_info 叠加。
- python/sglang/srt/model_executor/forward_batch_info.py（模块 前向批次；类别 source；类型 data-contract；符号 get_server_return_hidden_states_mode, get_required_capture_hidden_mode）：定义服务端模式解析与 required capture mode 聚合逻辑，是连接 server args 与图 runner 的枢纽。
- python/sglang/srt/server_args.py（模块 服务配置；类别 source；类型 core-logic；符号 _handle_return_hidden_states_mode）：新增 --return-hidden-states-mode 服务端参数

并维护与 --enable-return-hidden-states 的兼容映射，直接约束全链路可允许的最大捕获模式。

- test/registered/unit/model_executor/runner/test_hidden_state_graph_recapture.py（模块 图捕获测试；类别 test；类型 test-coverage；符号 TestHiddenStateGraphRecapture, test_server_mode_sets_graph_capture_ceiling, _make_runner, _make_forward_batch）：覆盖图捕获 ceiling 策略最完整的新增测试：server 模式映射、强弱模式复用、超限拒绝、spec override 与 prefill 图 fallback。
- test/registered/unit/managers/test_batch_result_processor_hidden_states.py（模块 结果处理测试；类别 test；类型 test-coverage；符号 _make_processor, _PrefillReq, init, finished）：针对 prefill 偏移与 decode 有界存储的精准回归：middle chunk 前进、mixed FULL/LAST 布局、spec 步长下的 finished_len 过滤。

关键符号：get_return_hidden_states_mode, get_request_return_hidden_states_mode, get_batch_return_hidden_states_mode, need_return_hidden_states, get_server_return_hidden_states_mode, get_required_capture_hidden_mode, _append_prefill_hidden_states, _append_decode_hidden_states, _get_prefill_hidden_capture_mode, _validate_capture_hidden_mode, _handle_return_hidden_states_mode, _normalize_return_hidden_states

关键源码片段

python/sglang/srt/managers/schedule_batch.py

定义 ReturnHiddenStatesMode 类型与 4 个模式映射函数，Req 与 ScheduleBatch 生命周期各环节全部改为按 CaptureHiddenMode 聚合，是本 PR 的数据契约核心。

```
ReturnHiddenStatesMode = Union[bool, Literal["last"]]
# 请求级 hidden-state 模式：False / True / "last" 分别映射到
# CaptureHiddenMode.NULL / FULL / LAST，规范化统一在这里完成。


def get_return_hidden_states_mode(
    return_hidden_states: ReturnHiddenStatesMode,
) -> CaptureHiddenMode:
    if return_hidden_states is True:
        return CaptureHiddenMode.FULL
    if return_hidden_states == "last":
        return CaptureHiddenMode.LAST
    if return_hidden_states is False:
        return CaptureHiddenMode.NULL
    raise ValueError(
        "return_hidden_states must be a boolean or the string literal 'last'."
    )


def get_request_return_hidden_states_mode(
    return_hidden_states: Union[List[ReturnHiddenStatesMode], ReturnHiddenStatesMode],
) -> CaptureHiddenMode:
```

```
    # 支持 per-prompt 列表：列表取 max，保证批内最强者决定捕获布局。
    if isinstance(return_hidden_states, list):
        return max(
            (get_return_hidden_states_mode(mode) for mode in return_hidden_states),
            default=CaptureHiddenMode.NULL,
        )
    return get_return_hidden_states_mode(return_hidden_states)


def get_batch_return_hidden_states_mode(reqs: List[Req]) -> CaptureHiddenMode:
    # 批级模式取各请求最大值：FULL 可覆盖 LAST/NULL，LAST 可覆盖 NULL，
    # 混合批量时图捕获与切片布局始终容纳最强者。
    mode = CaptureHiddenMode.NULL
    for req in reqs:
        mode = max(mode, req.return_hidden_states_mode)
    return mode


def need_return_hidden_states(
    return_hidden_states: Union[List[ReturnHiddenStatesMode], ReturnHiddenStatesMode],
) -> bool:
    return get_request_return_hidden_states_mode(return_hidden_states).need_capture()
```

python/sglang/srt/managers/scheduler_components/batch_result_processor.py

prefill 切片偏移与 decode 有界存储的实现地，集中体现了 radix cache 语义、inflight middle chunk 和 spec 越界三个易错点，是 review 讨论最密集的文件。

```
def _append_prefill_hidden_states(
    self,
    *,
    req: Req,
    logits_output: LogitsProcessorOutput,
    hidden_state_offset: int,
    capture_hidden_mode: CaptureHiddenMode,
    extend_input_len: int,
    store: bool = True,
) -> int:
    # 偏移必须按每个 batch 行无条件前进（包括 inflight 中间 chunk 与不存储的行），
    # 步长为 extend_input_len 而非 len(req.origin_input_ids)：
    # radix cache 命中时 tensor 仅含未缓存后缀，用整段长度会覆盖到
    # 下一个请求的行，导致 "last" 请求读到错误向量。
    if capture_hidden_mode.is_full():
        start = hidden_state_offset
        hidden_state_offset += extend_input_len
        if not store or not req.return_hidden_states:
            return hidden_state_offset
        req_hidden_states = logits_output.hidden_states[start:hidden_state_offset]
        if req.return_hidden_states is True:
```

```
            req.hidden_states.append(req_hidden_states.cpu().clone().tolist())
        elif req.return_hidden_states == "last":
            req.hidden_states.append(req_hidden_states[-1].cpu().tolist())
    elif capture_hidden_mode.is_last():
        index = hidden_state_offset
        hidden_state_offset += 1
        if store and req.return_hidden_states:
            req.hidden_states.append(
                logits_output.hidden_states[index].cpu().tolist()
            )
    else:
        raise ValueError(
            f"Unexpected hidden states capture mode: {capture_hidden_mode}"
        )
    return hidden_state_offset


@staticmethod
def _append_decode_hidden_states(
    *,
    req: Req,
    hidden_states: torch.Tensor,
    start: int,
    accept_len: int,
) -> None:
    if accept_len <= 0:
        return
    if req.return_hidden_states == "last":
        # "last" 模式用覆盖写入而非 extend，保证步数再多也只在
        # req.hidden_states 里保留一个向量（内存有界）。
        valid_accept_len = accept_len
        if req.finished_len is not None:
            step_start = len(req.output_ids) - accept_len
            valid_accept_len = max(
                0,
                min(accept_len, req.finished_len - step_start),
            )
        if valid_accept_len > 0:
            req.hidden_states[:] = [
                hidden_states[start + valid_accept_len - 1].cpu().tolist()
            ]
    else:
        req.hidden_states.extend(
            hidden_states[start : start + accept_len].cpu().tolist()
        )
```

python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py

图捕获策略重构的核心载体：删除 recapture_if_needed 的动态重捕获，改为启动时固定 server 级 capture ceiling 的静态校验，消除请求模式交替导致的图重建抖动。

```
# --- 图捕获模式初始化：启动即固定为 server 配置的最大模式 ---
# 旧实现：enable_return_hidden_states 为 True 时硬编码 FULL，导致第一个
# 普通 False 批次立刻触发一次全量重捕获；随后交替请求还会在
# FULL/LAST/NULL 之间反复清理并重建整批图。
self.capture_hidden_mode = self.return_hidden_states_mode


def _validate_capture_hidden_mode(self, forward_batch: ForwardBatch) -> None:
    # 新策略：图只按 server 上限捕获一次，弱模式请求直接复用，
    # 超限则在入口显式报错而不是静默重建图，避免运行时抖动。
    if self.capture_hidden_mode < forward_batch.capture_hidden_mode:
        raise RuntimeError(
            "The runtime hidden-state mode exceeds the fixed CUDA graph "
            f"capture mode ({self.capture_hidden_mode.name})."
        )


# load_batch 入口处调用（原 recapture_if_needed 的位置）：
# buffers = self.buffers
# self._validate_capture_hidden_mode(forward_batch)
```

评论区精华

核心交锋集中在 5 个问题上，全部在 reviewer 二审后 resolved：

- prefill 偏移语义：JustinTong0323 指出 _append_prefill_hidden_states 按 len(req.origin_input_ids) 前进偏移，radix cache 命中时 tensor 只含未缓存后缀，会导致下一个 "last" 请求读错行或空切片；作者改为按 extend_input_len_per_req[i] 步进，并新增暖 cache 混合模式数值一致性测试。
- 图捕获策略：Justin 要求避免在每次精确模式切换时重捕获全部图（启动即 FULL 时第一个 False 批次会立即重捕获 NULL，交替批次会频繁重建）；作者改为固定 server 级 capture ceiling，弱模式复用、超限显式拒绝，并给出 H20 上 21.9-24.5 ms 无重捕获的验证数据。
- EAGLE tc_piecewise 死图：Justin 指出 server 模式 "last" 时 legacy bool 绕过 skip guard，会复现 #28870 中 prefill 捕获与 target 需求不一致导致的 dead-graph 损坏 FP4/TRTLLM-MoE 问题；作者让 skip guard 依赖有效 capture mode。
- "last" 存储有界性：decode 原来仍 append 所有向量，长生成保留 O(tokens × hidden_size) 对象；新增 _append_decode_hidden_states 覆盖式写入，并断言 len(req.hidden_states) 恒为 1。
- io_struct 批量规范化：per-prompt 模式列表未随 parallel_sample_num 展开会在 n>1 时 IndexError；新增归一化函数覆盖校验、展开与广播。
  - prefill hidden-state 切片偏移与 radix cache 语义 (correctness): 改为按 extend_input_len_per_req[i] 步进，调度器在返回 hidden states 时保留 per-request

extend 长度；新增 warm-cache 混合模式测试，并断言 full 模式最后一行与 "last" 向量数值相等。

- CUDA 图捕获策略：从按需重捕获改为固定 server 级 ceiling (design): 改为启动时按 server 配置最大模式固定捕获一次，弱请求复用图，超限显式拒绝；新增 FULL->NULL->LAST->FULL 过渡与 decode/prefill/CPU 三 runner 的校验测试，H20 验证重复切换约 21.9-24.5 ms 无重捕获。
- EAGLE tc_piecewise 死图捕获与 #28870 前科 (correctness): skip guard 改为依赖有效 capture mode：server ceiling 低于 FULL 时跳过 EAGLE 不安全 prefill 图捕获，FULL 与 BCG 行为不变；新增 EAGLE target + tc_piecewise + server "last" 回归测试。
- hidden_state_offset 对 inflight middle chunk 也须前进 (correctness): 偏移改为对每个 batch 行无条件消费：FULL 模式 +extend_input_len、LAST 模式 +1，无论该行是否存储；新增 active middle chunk 后接新 "last" 请求在 FULL/LAST 两种布局下的回归，并在 H20 上对拍验证。
- "last" 模式 decode 存储有界性 (performance): 新增 _append_decode_hidden_states：last 模式用 finished_len 排除 spec 越界后覆盖 req.hidden_states[:] 为单向量；多步 spec 回归断言 len(req.hidden_states) 恒为 1。
- return_hidden_states 列表与 parallel sampling 的批量规范化 (correctness): 新增 _normalize_return_hidden_states：校验列表长度与每个模式、按 parallel_sample_num 展开、标量广播到展开后的批次；单元测试覆盖 n=2 的 [False, "last"] 展开、非法长度与非法模式。
- ServerArgs 程序化构造的模式校验 (correctness): _handle_return_hidden_states_mode 显式拒绝非法值并抛错，新增构造函数级回归测试验证非法模式被拒绝。

风险与影响

- 风险：
  1. prefill 偏移正确性：batch_result_processor.py 的偏移推进依赖 extend_input_len_per_req 在未开启 logprob 时仍被调度器保留（作者为此专门更新了 scheduler），任何后续改动若破坏该长度传递，"last" 会静默读到错误行，需保持测试覆盖。
  2. 图固定 ceiling 的行为变更：_validate_capture_hidden_mode 对超限模式抛 RuntimeError 是硬失败，现有依赖运行时动态开启 hidden states 的部署会直接报错而非自动重捕获；同时 FULL 模式启动即常驻 FULL 图，纯文本负载虽复用图但每步多出 hidden capture 的开销未量化。
  3. EAGLE / dflash 组合回归：prefill_cuda_graph_runner.py 的 skip guard 现在依赖有效 capture mode，is_dflash_family() 仍强制 FULL，两者叠加的边界情况（如 spec worker override 为 LAST 而 server 为 FULL）需要持续关注。
  4. 配置兼容性：--enable-return-hidden-states 语义变为等价 full，任何直接读该 bool 的第三方代码会误解 "last" 场景；ServerArgs 程序化构造的非法值校验虽已补上，但 CLI 与构造两条路径的校验一致性仍需维护。

5. spec decode 交互：decode last 模式用 finished_len 过滤 spec 越界，若 finished_len 与 output_ids 更新时序变化（如新的 spec 算法），取最后有效 accept 向量的逻辑需同步调整。 - 影响：调用方：Engine / OpenAI 请求新增 "last" 取值，响应从序列降为单向量，payload、序列化、网络与存储成本显著下降；支持 [False, True, "last"] 混合批量，并行采样 n>1 时模式列表自动展开。系统：CUDA/CPU 图从运行时动态重捕获变为启动时按 server 上限固定捕获，消除了请求模式交替导致的图重建抖动；批级 max 聚合使混合模式共享同一捕获布局。团队：新增服务端参数与兼容矩阵，文档与示例已同步；后续 pooled hidden states 等特性可直接复用 CaptureHiddenMode 语义，但也意味着 hidden states 相关配置需要集中维护。 - 风险标记：核心批处理路径变更，CUDA/CPU 图捕获策略重构，与 EAGLE tc_piecewise 交互（#28870 前科），新旧服务端参数兼容映射，radix cache 命中下的偏移语义

关联脉络

- PR #28870 (review 中引用的历史 PR) EAGLE dead-graph capture 损坏 FP4/TRTLLM-MoE decode replay: review 评论明确引用 #28870 说明 EAGLE target tc_piecewise 在错误 capture mode 下会形成 dead graph；本 PR 对 prefill_cuda_graph_runner skip guard 的修复正是为了避免回归该问题。
- PR #33168 Fix the chunked-prefix-cache gate writing config the backends never read: 同为 " 配置 / 请求模式写到正确位置 " 类问题：门控配置写错对象导致后端静默不读。本 PR 新增服务端 capture mode 参数，同样涉及 server_args 与后端读取的连通性，需保持这类配置的单一事实来源。
- PR #32223 [perf] Assemble flat prompt top logprobs scheduler-side as numpy arrays: 与本 PR 同路径（batch_result_processor / output_streamer / 调度器侧 meta 输出组装），在结果处理侧为输出载荷瘦身，与本 PR 降低 hidden states 载荷的目标同向，后续改动互相影响。