PR #32223 完整报告

sgl-project/sglang

[perf] Assemble flat prompt top logprobs scheduler-side as numpy arrays

合并时间：2026-08-01 09:00

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32223

执行摘要

- 一句话：调度器侧扁平化 prompt top logprobs 数组，两跳 IPC 省约 30ms
- 推荐动作：值得精读。三个设计点尤其值得借鉴：(1) 把数据面"组装前移 + 可选字段 + 默认 None"的兼容策略——新字段只对 opt-in 请求非 None，未使用方 wire 不变；(2) 共享校验原语 build_flat_input_top_logprobs_arrays 在 scheduler 与 tokenizer 两个进程复用，保证 fallback 语义一致；(3) 提交了 env-gated 的可复现序列化 microbench。需要留意的是 ch-wan 提出的缓存遮蔽与 idx 校验两个未解决的健壮性问题，未来若把 flat 格式扩展到 MIS 或其他请求路径，应先补齐这些防御。

功能与动机

PR body 明确指出：即便 #32078/#31960 已让 response 侧序列化变便宜，scheduler 仍把 prompt top logprobs 以嵌套的逐位置 Python 列表形式放在 BatchTokenIDOutput/BatchStrOutput 里跨两跳 ZMQ（scheduler → detokenizer → tokenizer manager）传输。实测数据：32768 位置、k=2 时嵌套形式 pickle.dumps 5.0ms + pickle.loads 10.3ms、1.12MB；扁平 f32/i32 数组仅 0.03ms + 0.03ms、0.53MB。作者估算每跳省约 15ms、两跳共约 30ms，且发生在服务器两个最忙的单例事件循环上。

实现拆解

按以下 5 步拆解：

1. 共享转换原语下沉到 io_struct：python/sglang/srt/managers/io_struct.py 新增 build_flat_input_top_logprobs_arrays，把嵌套行统一转换为 (float32 [rows,k], int32 [rows,k], null_prefix) 三元组，并校验"仅前导 null、每行长度均为 k"的矩形约束，不满足时抛 ValueError。同时 TokenizedGenerateReqInput 增加 return_flat_raw_top_logprobs 布尔位，随 GenerateReqInput → TokenizedGenerateReqInput → Req 一路传递（scheduler.py 构造 Req 时透传，schedule_batch.py 的 ReqLogprob 增加三个 flat 字段用于承载数组）。
2. scheduler 侧组装：python/sglang/srt/managers/scheduler_components/logprob_result_processor.py 在 add_input_logprob_return_values 的 relevant_tokens_len 断言之后新增 _flatten_input_top_logprobs，对 opt-in 请求调用共享原语；成功则清空嵌套行（后续 batch 输出只带数组），失败则保留嵌套格式并打 WARNING，语义与 tokenizer manager 既有 fallback 完全对齐。

3. IPC 线格式扩展：BatchTokenIDOutput/BatchStrOutput 增加 input_top_logprobs_val_flat / idx_flat / null_prefix 三个可选字段；output_streamer.py 在 _GenerationStreamAccumulator 中收集这三个字段，并仅在 batch 中存在 opt-in 请求时（has_input_top_logprobs_flat）才写入 payload，普通路径保持 None，从而对未使用方 wire 完全不变；multi_tokenizer_mixin.py 与 detokenizer_manager.py 分别按索引拆分 / 透传字段。
4. tokenizer 侧消费：tokenizer_manager.py 的 convert_logprob_style 把数组中转到 ReqState.input_top_logprobs_scheduler_flat（三者成组赋值）；add_logprob_to_meta_info 优先走新函数 _build_flat_input_top_logprobs_fields_from_arrays（b64 模式 tobytes() 后 base64，非 b64 模式 tolist()），并复用 input_top_logprobs_flat_fields 缓存；行版 _build_flat_input_top_logprobs_fields 改为复用同一原语做合法性检查，但非 b64 分支仍从原始行 flatten，以保持旧路径 JSON 数值的 float64 精度。
5. 测试配套：test/registered/unit/managers/test_flat_raw_top_logprobs.py 新增 28 个用例（+367 行），覆盖 flag on/off 行为差异、分块 prefill 与一次性 prefill 结果一致、参差行 fallback（assertLogs 验证告警）、from-arrays 与 from-rows 逐字段等价（flat / b64 / 全 null）、pickle 与 msgpack 双传输 roundtrip、默认 None wire 检查，并提交了一个环境变量门控的序列化 microbench。

关键文件：

- python/sglang/srt/managers/tokenizer_manager.py（模块 分词管理；类别 source；类型 core-logic；符号 _build_flat_input_top_logprobs_fields_from_arrays, add_logprob_to_meta_info, convert_logprob_style）：tokenizer 侧消费 scheduler 数组的核心：新增 _build_flat_input_top_logprobs_fields_from_arrays、ReqState.input_top_logprobs_scheduler_flat 状态，并在 add_logprob_to_meta_info / convert_logprob_style 中接入数组路径，同时保留嵌套回退。
- python/sglang/srt/managers/scheduler_components/logprob_result_processor.py（模块 概率输出；类别 source；类型 dependency-wiring；符号 _flatten_input_top_logprobs）：scheduler 侧组装入口：新增 _flatten_input_top_logprobs，在 prefill 长度断言后把嵌套行转数组并清空嵌套列表，是性能收益的起点。
- python/sglang/srt/managers/io_struct.py（模块 传输结构；类别 source；类型 core-logic；符号 build_flat_input_top_logprobs_arrays）：共享转换原语 build_flat_input_top_logprobs_arrays 与两个 batch 输出结构的新增可选字段定义，是整个数据面契约所在，两个进程共用同一套矩形性校验。
- python/sglang/srt/managers/scheduler_components/output_streamer.py（模块 输出流；类别 source；类型 core-logic）：负责把 Req 上的 flat 数组收集进 batch payload，并用 has_input_top_logprobs_flat 保证普通请求 wire 不变。
- test/registered/unit/managers/test_flat_raw_top_logprobs.py（模块 概率输出；类别 test；类型 test-coverage；符号 _make_logprob_processor, TestSchedulerFlatAssembly, test_flat_arrays_replace_nested_rows, test_chunked_matches_one_shot）：28 个新增测试用例，覆盖 scheduler 组装（flag on/off、chunked 与 one-shot 等价、参差 fallback）、tokenizer 两条路径逐字段等价、

pickle/msgpack roundtrip 与默认 None wire 检查，另有 env-gated 序列化 bench。

- python/sglang/srt/managers/schedule_batch.py（模块 请求批次；类别 source；类型 core-logic）：ReqLogprob 增加三个 flat 字段、Req.__init__ 接收并保存 return_flat_raw_top_logprobs，是请求级状态传递的底座。

关键符号：build_flat_input_top_logprobs_arrays, _flatten_input_top_logprobs, _build_flat_input_top_logprobs_fields_from_arrays, add_logprob_to_meta_info, convert_logprob_style

关键源码片段

python/sglang/srt/managers/tokenizer_manager.py

tokenizer 侧消费 scheduler 数组的核心：新增 _build_flat_input_top_logprobs_fields_from_arrays、ReqState.input_top_logprobs_scheduler_flat 状态，并在 add_logprob_to_meta_info / convert_logprob_style 中接入数组路径，同时保留嵌套回退。

```
def _build_flat_input_top_logprobs_fields_from_arrays(
    val_arr: np.ndarray,
    idx_arr: np.ndarray,
    null_prefix: int,
    return_b64: bool = False,
) -> Dict[str, Any]:
    """从 scheduler 侧组装好的 [rows, k] 数组构造 flat 响应字段。

    与行版 `_build_flat_input_top_logprobs_fields` 共享同一套字段语义
    (shape / null_prefix / flat / b64)，保证两条组装路径产出的 JSON
    结构逐字段一致。b64 模式用 dtype 标记字段，便于以后调整位宽。
    """
    fields: Dict[str, Any] = {}
    if return_b64:
        # native-endian 连续二进制 + base64; dtype 标记由客户端解释
        fields["input_top_logprobs_val_flat_b64"] = pybase64.b64encode(
            val_arr.tobytes()
        ).decode("utf-8")
        fields["input_top_logprobs_idx_flat_b64"] = pybase64.b64encode(
            idx_arr.tobytes()
        ).decode("utf-8")
        fields["input_top_logprobs_val_flat_b64_dtype"] = "float32"
        fields["input_top_logprobs_idx_flat_b64_dtype"] = "int32"
    else:
        # 非 b64: 展平为普通列表，与旧行版 JSON 结构保持一致
        fields["input_top_logprobs_val_flat"] = val_arr.reshape(-1).tolist()
        fields["input_top_logprobs_idx_flat"] = idx_arr.reshape(-1).tolist()
    fields["input_top_logprobs_shape"] = [val_arr.shape[0], val_arr.shape[1]]
    fields["input_top_logprobs_null_prefix"] = null_prefix
    return fields
```

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

scheduler 侧组装入口：新增 _flatten_input_top_logprobs，在 prefill 长度断言后把嵌套行转数组并清空嵌套列表，是性能收益的起点。

```
def _flatten_input_top_logprobs(self, req: Req) -> None:
    """把嵌套的 prompt top logprob 行替换为 flat 数组。

    仅对 opt-in 了 return_flat_raw_top_logprobs 的请求生效，使 batch
    输出只携带两个 ndarray，而不是 num_positions * k 个 Python 列表，
    从而压低 scheduler → detokenizer 这条 ZMQ 链路的序列化开销。
    """
    if req.logprob.top_logprobs_num <= 0:
        return
    try:
        (
            req.logprob.input_top_logprobs_val_flat,
            req.logprob.input_top_logprobs_idx_flat,
            req.logprob.input_top_logprobs_flat_null_prefix,
        ) = build_flat_input_top_logprobs_arrays(
            req.logprob.input_top_logprobs_val,
            req.logprob.input_top_logprobs_idx,
            req.logprob.top_logprobs_num,
        )
    except ValueError as e:
        # 无法用 (shape, null_prefix) 表达的行（例如多条目打分）：
        # 保留嵌套格式，与 tokenizer manager 侧的 fallback 语义一致。
        logger.warning(
            "Falling back to nested input top logprobs for rid=%s: %s",
            req.rid,
            e,
        )
        return
    # 组装成功后清空嵌套行，后续 batch 输出只携带数组；该调用发生在
    # prefill 长度校验之后，因此不会破坏 relevant_tokens_len 断言。
    req.logprob.input_top_logprobs_val = []
    req.logprob.input_top_logprobs_idx = []
```

python/sglang/srt/managers/io_struct.py

共享转换原语 build_flat_input_top_logprobs_arrays 与两个 batch 输出结构的新增可选字段定义，是整个数据面契约所在，两个进程共用同一套矩形性校验。

```
def build_flat_input_top_logprobs_arrays(
    input_top_logprobs_val: List[Optional[List[float]]],
    input_top_logprobs_idx: List[Optional[List[int]]],
    top_logprobs_num: int,
) -> Tuple[np.ndarray, np.ndarray, int]:
    """把嵌套的逐位置 top logprob 行转换为 flat 数组格式。
```

```
    返回 (float32 值数组 [rows, k], int32 token id 数组 [rows, k],
    null_prefix)。开头的 null 行计入 null_prefix 并从数组中剔除。
    行必须满足: 只有前导 null、且每行长度都为 k，否则抛 ValueError
    (例如 multi-item scoring 的稀疏 delimiter 行无法用 shape 表达)。
    """
    num_rows = len(input_top_logprobs_val)
    null_prefix = 0
    # 统计前导 null 行，这些位置没有 top logprobs，由 null_prefix 承载
    while null_prefix < num_rows and not input_top_logprobs_val[null_prefix]:
        null_prefix += 1
    val_rows = input_top_logprobs_val[null_prefix:]
    idx_rows = input_top_logprobs_idx[null_prefix:]
    # 空行集合时以 top_logprobs_num 作为 k，保证 shape 仍为 [0, k]
    k = len(val_rows[0]) if val_rows else top_logprobs_num
    for offset, row in enumerate(val_rows):
        if row is None or len(row) != k:
            # 中间空行或参差长度无法由 (shape, null_prefix) 表达，必须显式失败
            raise ValueError(
                "return_flat_raw_top_logprobs requires rectangular top logprob "
                f"rows with nulls only in the leading prefix; row {null_prefix + offset} "
                f"has {None if row is None else len(row)} entries (expected {k})."
            )
    val_arr = np.asarray(val_rows, dtype=np.float32).reshape(len(val_rows), k)
    idx_arr = np.asarray(idx_rows, dtype=np.int32).reshape(len(idx_rows), k)
    # 注意：目前只对 val 行做矩形性校验，idx 行默认与 val 行形态一致；
    # review 已指出该假设未加固，参差 idx 可能抛出非 ValueError 异常。
    return val_arr, idx_arr, null_prefix
```

评论区精华

review 中核心交锋如下：

> chatgpt-codex-connector[bot]（P2，正确性）：--enable-mis 场景下 delimiter-only 的稀疏行若恰好是矩形，会通过 build_flat_input_top_logprobs_arrays 的校验，随后嵌套行被清空，响应暴露 null_prefix=0、shape=[num_delimiters, k]，客户端会把它们错误重建为位置 0..N-1。
>
> sshleifer（作者回复）：已在 #32078（db2f53b3a2）于请求校验层直接拒绝 MIS + flat，理由是 multi_item_delimiter_indices 在 GenerateReqInput 上可见，入口处快速失败能同时保护 scheduler 侧组装与 tokenizer 侧 builder，而不是依赖只能“偶尔”生效的内部 null 检查。
>
> ch-wan（正确性建议）：add_logprob_to_meta_info 的 scheduler-flat 分支只在 flat_fields is None 时构建，但如果早先的流式 chunk 走过嵌套路径并缓存了空字段（num_rows == 0），后来 chunk 再设置 input_top_logprobs_scheduler_flat 时缓存已非 None，真实数组永远不会被编码，形成静默空 payload。建议在 convert_logprob_style 赋值时清空缓存。

> ch-wan（健壮性建议）：build_flat_input_top_logprobs_arrays 只校验 val 行，idx 行直接 reshape，若形态不匹配会抛出非 ValueError 的异常，逃出 _flatten_input_top_logprobs 的 catch；最终代码未加镜像校验。
>
> ch-wan（两个 nit）：b64 文档声称 little-endian 但 tobytes() 是 native endian；三个平行列表（val / idx / null_prefix）缺少越界防御。

最终 ch-wan 以 APPROVED 合入，后四条建议未在最终 patch 中看到对应修改，属于"讨论后接受的已知边界"。

- MIS + flat 组合会丢失 delimiter 位置映射 (correctness): sshleifer 在 #32078（db2f53b3a2）于请求校验层直接拒绝 MIS + flat，入口处快速失败同时保护 scheduler 与 tokenizer 两侧。
- scheduler-flat 缓存可能被嵌套路径早先缓存遮蔽，导致静默空 payload (correctness): 建议在 convert_logprob_style 赋值 scheduler_flat 时清空缓存；最终 patch 未见修改，属讨论后接受的已知边界。
- build_flat_input_top_logprobs_arrays 只校验 val 行，idx 行可抛非 ValueError 异常 (correctness): 最终代码仍只校验 val 行，建议未被采纳。
- b64 文档声称 little-endian 但编码为 native-endian (documentation): 作为 nit 提出，未在最终 patch 中看到修改。
- 三个平行列表（val / idx / null_prefix）缺少越界防御 (correctness): 建议用断言或降级处理；当前 streamer 保证三者成组追加，风险被接受。

风险与影响

- 风险：
  - 静默空 payload 风险（tokenizer_manager.py add_logprob_to_meta_info）：ch-wan 指出的缓存遮蔽问题未被修复——若流式时序出现"先走嵌套路径缓存空字段、后收到 scheduler 数组"的情况，input_top_logprobs_flat_fields 非 None 会跳过数组编码，响应缺 flat 字段且无异常。当前 streamer 通常 prefill 后首个流式 chunk 就发送 input logprobs，实际触发概率低，但属于静默失败。
  - idx 校验缺失（io_struct.py build_flat_input_top_logprobs_arrays）：只对 input_top_logprobs_val 做矩形校验，idx 行未校验，np.asarray(...).reshape(...) 可能抛 TypeError 或非预期 ValueError，而 _flatten_input_top_logprobs 只捕获 ValueError，其他异常会逃出批量结果循环。
  - 浮点精度差异：非 b64 的 JSON 数值经 scheduler 数组路径会做 float32 舍入，与旧路径 float64 列表并非逐位一致；作者论证 logprobs 本就以 float32 计算，属 no-op，但严格讲输出字节可能不同。
  - 路径不完整覆盖：session / EPD / PP 请求不传递 flag，降级为旧嵌套路径，响应一致但拿不到 IPC 收益。
  - 传输安全：默认 pickle 走 protocol-5 buffer、msgpack 路径经 msgspec 的 ndarray 钩子编码解码（routed_experts 先例），roundtrip 测试已覆盖；big-endian 主机上 b64 二进制为 native-endian，与文档描述存在偏差。

- 影响：对用户：return_flat_raw_top_logprobs / _b64 的响应字段结构不变，行为兼容；长 prompt + top logprobs 请求（如 32k 位置）可获得约 30ms 每请求的事件循环 CPU 节省与约减半的 IPC payload（1.12MB → 0.53MB）。对系统：收益集中在 detokenizer 与 tokenizer manager 两个最忙单例 loop，且只对 opt-in 请求生效；普通请求 wire 逐字节不变，回归面受控。对团队：组装与校验逻辑收敛到 build_flat_input_top_logprobs_arrays 单一原语，两个进程共享同一语义，后续调整 dtype 或扩展格式只需改一处；测试把两条组装路径的字段级等价性固化下来，降低未来改动风险。
- 风险标记：IPC 数据面变更，缓存遮蔽致静默空 payload，idx 校验缺失，异常类型逃逸风险，旧路径行为保持

关联脉络

- PR #31960 Flat raw prompt top logprobs response format (predecessor): 本 PR 的直接前置：flat raw top logprobs 响应格式与 tokenizer 侧行版组装由此引入，本 PR 在其基础上把组装前移到 scheduler。
- PR #32078 Flat raw prompt top logprobs tokenizer-side assembly + MIS validation (predecessor): PR body 明确引用：response 侧序列化由它做便宜，且其中 db2f53b3a2 在请求校验层拒绝 MIS + flat，是本 PR 讨论中 MIS 守卫的落点。
- PR #29799 support rust sglang server: 同属降低进程间数据面（scheduler/detokenizer/tokenizer）序列化开销的优化脉络，本 PR 是纯 Python 侧的同类努力。