PR #29656 完整报告

sgl-project/sglang

feat: make mm_inputs msgpack-native

合并时间：2026-08-21 05:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29656

执行摘要

- 一句话：mm_inputs 改 msgpack 原生序列化，剔除 PickleWrapper
- 推荐动作：值得精读。这是 SGLang IPC 全面 msgpack 化的重要节点，展示了 msgspec.Struct 与动态字段兼容、稳定 Ext 协议设计、长度前缀 buffer 布局的性能取舍。重点学习 _pack_buffer_ext 的序列化布局和 __setattr__ 的动态字段路由设计；同时关注 wire ID 稳定性对后续演进的影响。

功能与动机

issue #29465 将「让 mm_inputs 完全兼容 msgpack」列为 P0 任务：TokenizedGenerateReqInput 和 TokenizedEmbeddingReqInput 的 mm_inputs 字段仍是 Optional[PickleWrapper]，携带 pickled 的 MultimodalProcessorOutput，其中包含 torch.Tensor、np.ndarray、CudaIpcTensorTransportProxy 以及任意结构的 model_specific_data。PR body 明确指出：'This change sends the structure and unsupported leaf values through msgpack directly, which removes duplicate serialization and reduces scheduler IPC latency.' 目标是彻底去除该字段的 pickle 包埋，为 IPC 全面 msgpack 化扫清障碍。

实现拆解

1. 数据结构基础改造：python/sglang/srt/managers/schedule_batch.py 中 MultimodalDataItem 与 MultimodalProcessorOutput 从 @dataclass 改为 msgspec.Struct（kw_only=True, dict=True, array_like=True，并在 review 后补 weakref=True）；定义 MultimodalDataValue: TypeAlias = object 作为异构字段的统一边界；__post_init__ 把 hash 归一化为低 64 位，__setattr__ 把未声明属性自动写入 model_specific_data，从而保留旧代码中 item.audio_feature_lens = ... 这类动态赋值，避免 qwen2_audio 等处理器崩溃。
2. 请求类型与序列化接线：python/sglang/srt/managers/io_struct.py 中 mm_inputs 字段类型从 Optional[PickleWrapper] 改为 Optional[MultimodalProcessorOutput]，并从 wrap_pickle_fields / unwrap_pickle_fields 删除对应行；enc_hook / dec_hook / ext_hook 迁出到独立 util 模块。
3. 新增 Ext 编解码核心：新建 python/sglang/srt/utils/msgpack_utils.py，定义稳定 wire ID（array=1、torch tensor=2、np array=3、SHM=4、CUDA IPC=5）；普通 leaf 用 _pack_ext 递归 msgpack 编码；大 buffer 用 _pack_buffer_ext 的 [4 字节元数据长度][元

数据][原始字节] 布局，避免嵌套 Ext 造成额外拷贝；_to_msgpack_state /_from_msgpack_state 显式还原 tuple、torch.Size、dtype、device。_encode_shm_pointer_mm_data 委托 __getstate__，_encode_cuda_ipc_tensor_proxy只序列化 proxy_state 与 sync_data_meta 白名单。

4. 模型与处理器适配：evs_module.py 删除 EVSDataItem / VideoEVSDataItem 子类，把 thw_grids、pre_chunked_input_ids 放入 model_specific_data，断言改为 item.is_video() and key in item.model_specific_data；nano_nemotron_vl.py 改用 item.set(...) 写入；transformers_auto.py 保留 token_type_ids 为 tensor，避免违反类型注解导致 typed decoder 崩溃。
5. 测试与契约保障：test/registered/unit/managers/test_io_struct.py 新增 TestTokenizedReqInputMsgpack，覆盖 round-trip、动态字段、hash 归一化、weakref、CUDA IPC 状态、未知 Ext 容错等；test/registered/unit/multimodal/test_evs.py 增加 EVS 数据落位断言。验证结果：test_io_struct.py 45 passed，test_msgpack_ipc_roundtrip.py 7 passed。

关键文件：

- python/sglang/srt/utils/msgpack_utils.py（模块 序列化；类别 source；类型 core-logic；符号 _pack_ext, _unpack_ext, _pack_buffer_ext, _unpack_buffer_ext）：新增核心序列化模块，定义稳定 Ext wire ID 与 buffer 布局，是本次变更的技术底座
- python/sglang/srt/managers/schedule_batch.py（模块 批处理结构；类别 source；类型 core-logic；符号 MultimodalDataItem, MultimodalProcessorOutput, post_init, getattr）：多模态数据结构从 dataclass 改为 msgspec.Struct，是本次变更的核心数据结构，包含动态字段路由与 hash 归一化
- python/sglang/srt/managers/io_struct.py（模块 请求结构；类别 source；类型 data-contract；符号 TokenizedGenerateReqInput, TokenizedEmbeddingReqInput, wrap_pickle_fields, unwrap_pickle_fields）：请求类型 mm_inputs 从 PickleWrapper 改为真实类型，并移除 wrap/unwrap pickle 逻辑
- test/registered/unit/managers/test_io_struct.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestTokenizedReqInputMsgpack, _make_mm_inputs, _round_trip, test_generate_mm_inputs_round_trip_without_pickle_wrapper）：新增 mm_inputs msgpack round-trip 测试套件，覆盖动态字段、hash 归一化、weakref、CUDA IPC 状态等回归场景
- python/sglang/srt/multimodal/evs/evs_module.py（模块 EVS 模块；类别 source；类型 refactor；符号 EVSDataItem, VideoEVSDataItem, redistribute_pruned_frames_placeholders, evs_video）：EVS 子类数据结构迁移到 model_specific_data，适配 msgpack 原生结构
- python/sglang/srt/models/nano_nemotron_vl.py（模块 模型适配；类别 source；类型 data-contract；符号 pad_input_ids）：EVS 字段写入方式改为 item.set 并检查 model_specific_data，保证跨进程数据契约
- python/sglang/srt/multimodal/processors/transformers_auto.py（模块 处理器；类别 source；类型 bugfix）：修复 token_type_ids 类型违约，避免 typed decoder 崩溃

- test/registered/unit/multimodal/test_evs.py（模块 EVS 测试；类别 test；类型 test-coverage；符号 test_evs_items_store_wire_data_in_model_specific_data）：新增 EVS 数据存放在 model_specific_data 的回归测试

关键符号：_pack_ext, _unpack_ext, _pack_buffer_ext, _unpack_buffer_ext, _to_msgpack_state, _from_msgpack_state, _restore_torch_tensor, MultimodalDataItem.post_init, MultimodalDataItem.getattr, MultimodalDataItem.setattr, MultimodalProcessorOutput.set, TokenizedGenerateReqInput.wrap_pickle_fields, pad_input_ids

关键源码片段

python/sglang/srt/utils/msgpack_utils.py

新增核心序列化模块，定义稳定 Ext wire ID 与 buffer 布局，是本次变更的技术底座

```
# 稳定的 wire ID，修改需同步 golden-wire 测试
_MSGPACK_EXT_ARRAY = 1
_MSGPACK_EXT_TORCH_TENSOR = 2
_MSGPACK_EXT_NP_ARRAY = 3
_MSGPACK_EXT_SHM_POINTER_MM_DATA = 4
_MSGPACK_EXT_CUDA_IPC_TENSOR_PROXY = 5
# 长度前缀元数据 head：4 字节大端长度
_MSGPACK_BUFFER_METADATA_SIZE = struct.Struct('>I')


def _pack_ext(code: int, obj: object) -> msgspec.msgpack.Ext:
    # 递归编码 object，内部 unsupported leaf 再走 enc_hook
    return msgspec.msgpack.Ext(code, msgspec.msgpack.encode(obj, enc_hook=enc_hook))


def _unpack_ext(data: memoryview) -> object:
    # Ext payload 仍是 msgpack，递归解码恢复类型
    return msgspec.msgpack.decode(data, ext_hook=ext_hook)


def _pack_buffer_ext(code: int, metadata: object, raw_data: memoryview) -> msgspec.msgpack.
Ext:
    # 布局：[metadata 长度 (4B)][metadata msgpack][raw bytes]
    # 避免把大块数据再包一层 msgpack，显著降低拷贝开销
    metadata_bytes = msgspec.msgpack.encode(metadata)
    payload = bytearray(_MSGPACK_BUFFER_METADATA_SIZE.pack(len(metadata_bytes)))
    payload.extend(metadata_bytes)
    payload.extend(raw_data)
    return msgspec.msgpack.Ext(code, payload)


def _unpack_buffer_ext(data: memoryview) -> tuple[object, memoryview]:
    if len(data) < _MSGPACK_BUFFER_METADATA_SIZE.size:
        raise msgspec.DecodeError('MessagePack buffer extension is missing metadata')
```

```
    (metadata_size,) = _MSGPACK_BUFFER_METADATA_SIZE.unpack_from(data)
    raw_data_offset = _MSGPACK_BUFFER_METADATA_SIZE.size + metadata_size
    if raw_data_offset > len(data):
        raise msgspec.DecodeError('MessagePack buffer extension has invalid metadata')
    metadata = msgspec.msgpack.decode(data[_MSGPACK_BUFFER_METADATA_SIZE.size:raw_
    data_offset])
    return metadata, data[raw_data_offset:]


def _to_msgpack_state(obj: object) -> object:
    # 显式把 tuple / torch.Size / dtype / device 转成带标记 dict，保留运行时类型
    if isinstance(obj, torch.dtype):
        return {'__torch_dtype__': _torch_dtype_name(obj)}
    if isinstance(obj, torch.device):
        return {'__torch_device__': str(obj)}
    if isinstance(obj, np.dtype):
        return {'__np_dtype__': obj.str}
    if isinstance(obj, torch.Size):
        return {'__torch_size__': list(obj)}
    if isinstance(obj, tuple):
        return {'__tuple__': [_to_msgpack_state(v) for v in obj]}
    if isinstance(obj, dict):
        return {k: _to_msgpack_state(v) for k, v in obj.items()}
    if isinstance(obj, list):
        return [_to_msgpack_state(v) for v in obj]
    return obj
```

python/sglang/srt/managers/schedule_batch.py

多模态数据结构从 dataclass 改为 msgspec.Struct，是本次变更的核心数据结构，包含动态字段路由与 hash 归一化

```
# Msgpack 原生容器 + Ext 解码的 tensor/transport leaf。
# 注意：泛型 tuple 会按 msgpack 语义解码为 list，因此 offsets 显式标注为 tuple 对。
MultimodalDataValue: TypeAlias = object


class MultimodalDataItem(msgspec.Struct, kw_only=True, dict=True, array_like=True):
    """单个多模态输入（一张图、一段视频或一段音频）的预处理产物。"""

    modality: Modality
    hash: Optional[int] = None
    pad_value: Optional[int] = None
    offsets: Optional[List[Tuple[int, int]]] = None
    format: MultimodalInputFormat = MultimodalInputFormat.NORMAL
    feature: Optional[MultimodalDataValue] = None
    precomputed_embeddings: Optional[MultimodalDataValue] = None
    # 处理器自定义字段：tensor/array/ 标量 / 传输代理统一走 object 边界
    model_specific_data: Dict[str, MultimodalDataValue] = msgspec.field(default_factory=dict)
```

```
    def __post_init__(self) -> None:
        # 归一化到低 64 位，避免 msgpack 拒绝超范围 int（如 md5/uuid4）
        if self.hash is not None:
            msgspec.Struct.__setattr__(self, 'hash', self.hash & _MM_HASH_MASK)

    def __getattr__(self, name: str) -> MultimodalDataValue:
        # 兼容 dataclass 时代通过属性访问 model_specific_data 的写法
        if name in self.model_specific_data:
            return self.model_specific_data[name]
        raise AttributeError(f'{type(self).__name__} object has no attribute {name}')

    def __setattr__(self, name: str, value: MultimodalDataValue) -> None:
        # 未声明字段自动落入 model_specific_data，保证跨进程不丢动态赋值
        if name in self.__struct_fields__:
            if name == 'hash' and isinstance(value, int) and not (0 <= value <= _MM_HASH_MASK):
                value &= _MM_HASH_MASK
            msgspec.Struct.__setattr__(self, name, value)
        else:
            self.model_specific_data[name] = value

    def set(self, key: str, value: MultimodalDataValue) -> None:
        self.model_specific_data[key] = value
```

test/registered/unit/managers/test_io_struct.py

新增 mm_inputs msgpack round-trip 测试套件，覆盖动态字段、hash 归一化、weakref、CUDA IPC 状态等回归场景

```
class TestTokenizedReqInputMsgpack(unittest.TestCase):
    def _make_mm_inputs(self, device='cpu'):
        # 覆盖典型 VLM 预处理产物：tensor、np.ndarray、np 标量、tuple、list
        return MultimodalProcessorOutput(
            mm_items=[
                MultimodalDataItem(
                    modality=Modality.IMAGE,
                    offsets=[(0, 1)],
                    format=MultimodalInputFormat.NORMAL,
                    feature=torch.tensor([[1.0, 2.0]], dtype=torch.float32, device=device),
                    model_specific_data={
                        'image_grid_thw': torch.tensor([[1, 1, 2]], dtype=torch.int64, device=device),
                        'patch_counts': np.array([2], dtype=np.int32),
                        'names': ['image0'],
                        'count': np.int64(2),
                        'enabled': np.bool_(True),
                        'size': (336, 336),
                    },
                )
            ],
            input_ids=[1, 2],
            padded_input_ids=[10, 10],
```

```
            im_token_id=10,
            mrope_positions=torch.tensor([[0, 1]], dtype=torch.int64, device=device),
            token_type_ids=torch.tensor([0, 0], dtype=torch.int64, device=device),
        )

    def _round_trip(self, req):
        req.wrap_pickle_fields() # 兼容旧 pickled 字段，mm_inputs 已不在其中
        decoded = msgpack_decode(msgpack_encode(req))
        decoded.unwrap_pickle_fields()
        return decoded

    def test_generate_mm_inputs_round_trip_without_pickle_wrapper(self):
        decoded = self._round_trip(
            TokenizedGenerateReqInput(
                input_text='', input_ids=array('q', [1, 2]), input_embeds=None,
                mm_inputs=self._make_mm_inputs(), token_type_ids=[0, 0],
                sampling_params=SamplingParams(), return_logprob=False,
                logprob_start_len=0, top_logprobs_num=0, token_ids_logprob=None,
                stream=False,
            )
        )
        self.assertIsInstance(decoded.mm_inputs, MultimodalProcessorOutput)
        item = decoded.mm_inputs.mm_items[0]
        self.assertEqual(item.modality, Modality.IMAGE)
        self.assertEqual(item.offsets, [(0, 1)]) # typed decoder 恢复 tuple 对
        self.assertTrue(torch.equal(item.feature, torch.tensor([[1.0, 2.0]], device='cpu')))
        # 动态字段经 __setattr__ 留在 model_specific_data
        self.assertTrue(torch.equal(
            item.model_specific_data['image_grid_thw'],
            torch.tensor([[1, 1, 2]], dtype=torch.int64, device='cpu'),
        ))
```

评论区精华

- alexnails 指出动态字段丢失的 BLOCKER：qwen2_audio.py / midashenglm.py 直接给 item 赋值，迁移后会触发 AttributeError 或静默降级。作者通过 __setattr__ 将未声明字段路由进 model_specific_data 并补测试。
- alexnails 指出 offsets tuple/list 混叠：msgpack 无 tuple 类型，typed decoder 会把 [(0,5)] 变成 [[0,5]]，导致 flatten_nested_list 展开后无法解包。作者将 offsets 注解改为 List[Tuple[int, int]]，利用 typed decoder 恢复 tuple 对。
- alexnails 指出 weakref.finalize 失败：dict=True 不提供弱引用，PD encode-disagg 会崩溃并泄漏 GPU embedding 槽。作者增加 weakref=True 并添加回归测试。
- alexnails 指出 hash 溢出：md5/sha256 的 128/256 位 int 或 uuid4().int 会让 msgpack 抛 OverflowError。作者归一化为低 64 位，保留 pad 计算语义。
- 围绕 DLPack 的设计讨论：作者解释 DLPack capsule 携带进程内指针，不满足跨进程 wire format，因此保留 SHM/CUDA IPC 代理作为数据面，raw tensor Ext 仅作为拷贝回

退。

- 性能讨论：嵌套 Ext 递归编码 8 MiB NumPy 耗时 8.19 ms，改用长度前缀 buffer ext 后降至 0.630 ms（约 13 倍）。
- merrymercy 要求把工具函数拆出 io_struct.py 并禁止 import alias，最终迁入 utils/msgpack_utils.py。
- tuple/torch.Size 序列化后类型丢失 (correctness): 作者在 7238d0ed 修复，在 state 中标记 tuple/ __torch_size__并显式重建，同时补充 CUDA IPC 回归。
- 动态字段赋值丢失导致处理器崩溃 (correctness): 作者用 __setattr__将未声明字段路由进 model_specific_data，并添加直接赋值 round-trip 测试。
- offsets 的 tuple/list 混叠导致批量解码错误 (correctness): 将 offsets 注解为 List[Tuple[int, int]]，typed decoder 自动恢复 tuple 对，round-trip 测试断言 [(0, 1)]。
- MultimodalProcessorOutput 缺少 weakref 支持 (correctness): 在 Struct 上加 weakref=True，并新增弱引用回归测试。
- hash 超 64 位导致 msgpack OverflowError (correctness): 在构造和赋值时归一化为低 64 位，保留模 2^30 pad 语义，避免改变缓存键。
- 嵌套 Ext 编码性能与 DLPack 取舍 (performance): 采用 [4-byte metadata length][metadata][raw bytes] 的 buffer ext；保留 DLPack 为不可行方案。
- transformers_auto 违反 token_type_ids 类型注解 (correctness): 去掉 .tolist() 保留 tensor，由 tokenizer manager 在 decode 后转 list；补 tensor round-trip 断言。
- 工具函数应拆出 io_struct.py (design): 提交 c8883b1716 将 hooks 迁到 sglang.srt.utils.msgpack_utils，并让请求类型检查跳过导入类。

风险与影响

- 风险：
  - wire 协议稳定性：Ext ID（1-5）是稳定契约，改动需同步 golden-wire 测试；跨版本滚升需考虑旧 pickle 请求与新版调度器的兼容。
  - 动态字段依赖 __setattr__ 路由：若处理器绕过 Struct 机制（如直接写 __dict__）或使用 msgspec.Struct.__setattr__，字段仍可能丢失，需持续排查各处理器。
  - CUDA IPC 生命周期：解码后的代理需正确初始化 _consumer_acknowledged 等状态，否则可能影响引用计数与资源释放。
  - 性能兜底：raw tensor Ext 是拷贝回退路径，大 payload 下仍高于描述符方案；未来若 profiling 显示瓶颈，需引入 auxiliary frames 或阈值切换。
  - 回归风险：核心 IPC 路径变更，涉及多模态请求从 tokenizer 到 scheduler 的全链路，CI 中多模态 e2e 测试必须全绿。
- 影响：
  - 对用户：多模态请求 IPC 延迟降低，pickle 不再用于 mm_inputs，消除了 pickle 带来的安全与兼容性风险。
  - 对系统：调度器接收到的 mm_inputs 是类型化的 msgpack 结构，去除了重复序列化；为后续全面去除 PickleWrapper、翻转默认 msgpack 传输铺路。

- 对团队：建立了可复用的 Ext 编解码模式，evs_module 等代码去子类化，统一了多模态数据契约，后续 IPC 迁移可复用相同模式。
- 影响范围：涉及 io_struct.py、schedule_batch.py、evs 模块、transformers_auto 处理器等 9 个文件，属于调度与多模态核心链路。
- 风险标记：核心 IPC 路径变更，wire 协议稳定性依赖 Ext ID，动态字段依赖 __setattr__ 路由，pickle 兼容性（旧数据），CUDA IPC 生命周期状态

关联脉络

- PR #28688 Convert IPC dataclasses to msgspec.Struct: issue #29465 中提及的基础迁移 PR，引入了 opt-in msgpack 传输；本 PR 在其基础上继续消除 mm_inputs 的 PickleWrapper。