PR #35718 完整报告

sgl-project/sglang

Support mxfp8 KV cache in PD transfer

合并时间：2026-08-21 13:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35718

执行摘要

- 一句话：MXFP8 KV 缓存打通 PD 分离式传输
- 推荐动作：值得精读。三个最值得关注的设计决策：1）scale 缓冲区为什么必须走 multi-component state channel 而不是追加到 KV lists —— 四个按长度推导几何关系的调用点决定了 append 会静默错位；2）interleaved / flat 两种布局对 item_len 语义的差异化处理（leading axis 一页一行 vs 按 slot 一页 page_size 行）；3）review 中从 MXFP8_SCALE 泛化到 BLOCK_SCALE 的命名决策，体现对 microscaling 生态（MXFP4、NVFP4）的前瞻。建议与前置依赖 #35689 一起阅读，理解 hybrid 模型 PD 的完整链路。

功能与动机

PR body 明确：MXFP8 KV 此前被 PD 分离式传输整体拒绝，get_contiguous_buf_infos 直接抛 KV transfer / disaggregation is unsupported for MXFP8 KV cache (scale buffers are not exposed)。目标就是暴露 scale buffers，让 hybrid attention + linear-attention 模型可以在 --kv-cache-dtype mxfp8 下服务 PD。PR body 还解释了关键约束：scale 缓冲区不能进入 KV lists，因为四个调用点（setup_state_kv_args 的 len(kv_data_ptrs) // layer_num、staging handler 的 len(kv_item_lens) // 2、prefill.py 的 layer range、common/conn.py 的 per-token bytes）都从数组长度推导几何关系，追加缓冲区会让它们全部静默错位；并且需要两个组件分别承载整序列与滑动窗口的 scale 索引，同时依赖已合入 main 的 #35689（空 mamba temporal buffer 的 batch 注册修复）。

实现拆解

1. 暴露 scale 访问器（python/sglang/srt/mem_cache/memory_pool.py）：移除 get_contiguous_buf_infos 的 NotImplementedError，新增 get_kv_scale_buf_infos 返回 (ptrs, lens, item_lens)，并按布局区分 item_len 语义 —— interleaved（fa4 的 page_size == 128 路径）把页放在 leading axis，一行即一页；flat 布局按 slot 索引，一页跨 page_size 行。同时在 _create_buffers 覆写末尾用 _build_kv_buffer_descs 补齐 _kv_buffer_descs，修复继承访问器的 AttributeError。
2. 扩展状态协议（base/conn.py、swa_memory_pool.py）：StateType 新增 BLOCK_SCALE / BLOCK_SCALE_SWA 两个组件类型（review 后从 MXFP8_SCALE 泛化改名）；SWAKVPool 新增 get_kv_scale_buf_infos 与 get_swa_kv_scale_buf_infos，分别委托给内部嵌套的两个 MXFP8 子池。

3. 状态通道接线（disaggregation/utils.py 的 setup_state_kv_args）：对 MHATokenToKVPoolMXFP8 追加 BLOCK_SCALE 组件；只有具体实现 SWAKVPool 且其 full_kv_pool 为 MHATokenToKVPoolMXFP8 时，才同时追加 BLOCK_SCALE 与 BLOCK_SCALE_SWA 两个组件 —— 其他 BaseSWAKVPool 实现按条目描述状态，不能复用该子池委托。
4. 传输侧 payload（decode.py、prefill.py、mooncake/conn.py）：把 _dsa_payload 重命名为语义更准确的 _full_kv_pages_payload，并让 BLOCK_SCALE 复用整序列页索引、BLOCK_SCALE_SWA 复用 _swa_payload 的窗口页索引；两端 payload 必须 positionally 一致，Mooncake 连接层同步适配新增组件的 per-token 字节处理。
5. 测试配套：新增 test/registered/unit/mem_cache/test_mxfp8_scale_transfer_buffers.py（注册 CPU CI），用 object.__new__构造只含访问器所需字段的池 stub，分别断言 interleaved 的 item_len 等于一行、flat 的 item_len 等于 page_size 行、lens // item_lens 等于页数，弥补 flat 布局无真实硬件路径的覆盖缺口；CI 反复重跑 disaggregation 套件（hybrid attention、dsv4、kimi linear、minimax sparse、wire、basic）均通过。

关键文件：

- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 get_contiguous_buf_infos, get_kv_scale_buf_infos）：核心改动：移除 get_contiguous_buf_infos 对 MXFP8 的硬拒绝，新增 get_kv_scale_buf_infos 暴露 UE8M0 scale 缓冲区 ptrs/lens/item_lens，并按 interleaved/flat 布局区分 item_len 语义；同时在 _create_buffers 覆写末尾补齐 _kv_buffer_descs，修复继承访问器的 AttributeError。
- python/sglang/srt/disaggregation/utils.py（模块 状态接线；类别 source；类型 dependency-wiring；符号 setup_state_kv_args）：setup_state_kv_args 是状态通道的装配点：为 MHATokenToKVPoolMXFP8 追加 BLOCK_SCALE 组件，为具体 SWAKVPool 追加 BLOCK_SCALE 与 BLOCK_SCALE_SWA 两个组件，决定 scale 缓冲区如何随 KV 一起走 wire，是整个方案的接线枢纽。
- test/registered/unit/mem_cache/test_mxfp8_scale_transfer_buffers.py（模块 单元测试；类别 test；类型 test-coverage；符号 _pool, TestMXFP8ScaleTransferBuffers, test_interleaved_item_len_is_one_page_row, test_flat_item_len_covers_a_whole_page）：新增单元测试，用极小池 stub 验证 interleaved（item_len 一行）与 flat（item_len 一页 page_size 行）两种布局的 scale 缓冲区几何，并断言 lens // item_lens 等于页数，弥补 flat 布局无真实硬件的覆盖缺口。
- python/sglang/srt/disaggregation/base/conn.py（模块 传输协议；类别 source；类型 core-logic；符号 StateType）：StateType 枚举新增 BLOCK_SCALE / BLOCK_SCALE_SWA 两个状态组件类型，是 wire 协议的语义定义；review 中关于泛化命名的讨论在这里落地。
- python/sglang/srt/disaggregation/decode.py（模块 PD 传输；类别 source；类型 core-logic；符号 _dsa_payload, _full_kv_pages_payload）：decode 侧把 _dsa_payload 重命名为 _full_kv_pages_payload，并注册 BLOCK_SCALE、BLOCK_SCALE_SWA 两个新状态的页索引载荷，保证 scale 与 KV 数据页一一对应。

- python/sglang/srt/disaggregation/prefill.py（模块 PD 传输；类别 source；类型 core-logic；符号 _dsa_payload, _full_kv_pages_payload）：prefill 侧与 decode.py 对称：整序列 scale 走 _full_kv_pages_payload，滑动窗口 scale 走 _swa_payload，两端索引必须 positionally 一致。
- python/sglang/srt/mem_cache/swa_memory_pool.py（模块 滑动窗口池；类别 source；类型 core-logic；符号 get_kv_scale_buf_infos, get_swa_kv_scale_buf_infos）：SWAKVPool 增加 get_kv_scale_buf_infos 与 get_swa_kv_scale_buf_infos，把内部嵌套的 MXFP8 子池 scale 信息委托暴露给状态通道。
- python/sglang/srt/disaggregation/mooncake/conn.py（模块 传输协议；类别 source；类型 core-logic）：Mooncake 传输连接按状态组件处理 per-token 字节数，新增组件需要在此对齐，否则 RDMA 传输量计算与数据切分会错位。

关键符号：get_kv_scale_buf_infos, get_swa_kv_scale_buf_infos, setup_state_kv_args, _full_kv_pages_payload, get_contiguous_buf_infos

关键源码片段

python/sglang/srt/mem_cache/memory_pool.py

核心改动：移除 get_contiguous_buf_infos 对 MXFP8 的硬拒绝，新增 get_kv_scale_buf_infos 暴露 UE8M0 scale 缓冲区 ptrs/lens/item_lens，并按 interleaved/flat 布局区分 item_len 语义；同时在 _create_buffers 覆写末尾补齐 _kv_buffer_descs，修复继承访问器的 AttributeError。

```
# python/sglang/srt/mem_cache/memory_pool.py

# 下面这段位于 MXFP8 池对 _create_buffers 的整体覆写末尾。
# 基类只在被替换方法的末尾构建 PD 传输描述符，这里需要手动补齐，
# 否则继承的访问器会抛 AttributeError。
self._kv_buffer_descs = self._build_kv_buffer_descs()


def get_kv_scale_buf_infos(self):
    """返回 (ptrs, lens, item_lens)，按 k 后 v 排列，类型为 UE8M0。

    interleaved 布局把页放在 leading axis，一页的 scale 是连续一行；
    flat 布局按 slot 索引，一页跨 page_size 行。
    """
    tensors = self.k_scale_buffer + self.v_scale_buffer
    ptrs = [t.data_ptr() for t in tensors]
    lens = [t.nbytes for t in tensors]
    row_bytes = [t[0].nbytes for t in tensors]
    if self.mxfp8_sf_interleaved:
        item_lens = row_bytes
    else:
        item_lens = [rb * self.page_size for rb in row_bytes]
    return ptrs, lens, item_lens
```

python/sglang/srt/disaggregation/utils.py

setup_state_kv_args 是状态通道的装配点：为 MHATokenToKVPoolMXFP8 追加 BLOCK_SCALE 组件，为具体 SWAKVPool 追加 BLOCK_SCALE 与 BLOCK_SCALE_SWA 两个组件，决定 scale 缓冲区如何随 KV 一起走 wire，是整个方案的接线枢纽。

```
# python/sglang/srt/disaggregation/utils.py

from sglang.srt.mem_cache.memory_pool import (
    DSATokenToKVPool,
    HybridLinearKVPool,
    MHATokenToKVPoolMXFP8,
    MiniMaxSparseKVPool,
)
from sglang.srt.mem_cache.swa_memory_pool import SWAKVPool

# 纯 MXFP8 池：整序列的 block scale 作为独立组件走 state channel，
# 不能塞进 KV lists —— 否则四个按长度推导几何关系的调用点会全部错位。
if isinstance(token_to_kv_pool, MHATokenToKVPoolMXFP8):
    append_state_component(
        kv_args,
        StateType.BLOCK_SCALE,
        *token_to_kv_pool.get_kv_scale_buf_infos(),
    )

# 混合池：SWAKVPool 内部嵌套两个 MXFP8 子池，full-attention 的 scale
# 跟随整个序列，滑动窗口的 scale 需要 translate_loc_from_full_to_swa 的
# 窗口索引，因此拆成两个组件，各自继承所属 KV 的索引载荷。
if isinstance(token_to_kv_pool, SWAKVPool) and isinstance(
    token_to_kv_pool.full_kv_pool, MHATokenToKVPoolMXFP8
):
    append_state_component(
        kv_args,
        StateType.BLOCK_SCALE,
        *token_to_kv_pool.get_kv_scale_buf_infos(),
    )
    append_state_component(
        kv_args,
        StateType.BLOCK_SCALE_SWA,
        *token_to_kv_pool.get_swa_kv_scale_buf_infos(),
    )
```

评论区精华

> zcnrex（base/conn.py）：建议把 MXFP8_SCALE 泛化，附上 OCP MX 格式对比表 —— MXFP8 E4M3/E5M2 为 32 元素 block + 8 bit E8M0 scale，MXFP4 同 block 32，NVFP4 为 block 16 + per-tensor FP32，提议改为 block_scale / block_scale_swa / tensor_scale，为 microscaling 数据类型预留扩展。

> ispobock：make sense。随后以提交 rename to block scale state types 落地为 BLOCK_SCALE / BLOCK_SCALE_SWA。
>
> zcnrex 最终 APPROVED：LGTM!

- StateType 命名泛化：MXFP8_SCALE 改为 BLOCK_SCALE (design): ispobock 回复 make sense，随后以提交 rename to block scale state types 落地为 StateType.BLOCK_SCALE 与 StateType.BLOCK_SCALE_SWA，zcnrex APPROVED（LGTM!）。

风险与影响

- 风险：
  1. 索引语义敏感（正确性）：BLOCK_SCALE_SWA 与 SWA 组件共用 _swa_payload，scale 页索引必须与 KV 数据页索引严格对齐；PR body 自证「把窗口组件指向整序列 payload 会退化为空或截断输出」，任何窗口对齐偏差都会造成反量化读取错位。
  2. wire 协议跨版本兼容：StateType 枚举新增两个值、state channel 组件数量变化，prefill 与 decode 两端版本不一致会在 wire 上静默错位（组件按序解析），升级需保证两端同版本。
  3. flat 布局无硬件覆盖：fa4 将 page_size 固定为 128，真实硬件只走 interleaved 分支；flat 布局仅有单元测试，真实索引路径未验证。
  4. 性能开销：E8M0 scale 每 32 元素附加 1 字节，RDMA payload 轻微增大；评测中 MXFP8 比 bf16 多约 7% completion tokens（822k vs 768k），长序列更容易触达 max_tokens 上限导致截断。
  5. 残留限制：set_kv_buffer_prefix_valid 仍对 MXFP8 拒绝，DSpark / DFlash speculative 与 PD 组合场景仍不可用。 - 影响：该 PR 直接影响 PD 分离式推理的可用模型范围：hybrid attention + linear-attention 模型（如 DeepSeek 系）现在可以 --kv-cache-dtype mxfp8 服务 PD，在相近精度（AIME26 score 0.9333、stop_rate 0.9667 vs bf16 无截断）下换取显存收益。改动横跨 mem_cache（内存池布局语义）与 disaggregation（状态通道 payload）两个子系统，共 8 个文件、约 121 行。对团队而言，state channel 从 KV 相邻专用缓冲走向通用 microscaling block scale 组件，为未来 TENSOR_SCALE（NVFP4 等）预留了扩展点，但也要求后续所有传输组件都遵循「独立组件 + 各自索引载荷」的约定。 - 风险标记：PD 传输核心路径变更，state 索引语义敏感，flat 布局无硬件覆盖，wire 跨版本兼容，残留 prefix-valid 限制

关联脉络

- PR #27770 [P/D disagg] Decode-side radix cache for SWA hybrid models (unified radix tree): 同一功能线：SWA hybrid 模型的 PD 传输。本 PR 的 BLOCK_SCALE_SWA 正是为 SWA 子池的 MXFP8 scale 补齐 PD 传输语义，与 #27770 的 decode 侧 radix 缓存复用相辅相成。
- PR #35081 Using unified radix tree by default for all case: mem_cache 池注册与状态组件体系持续演进的另一环；本 PR 依赖同一批池类型（SWAKVPool、MHATokenToKVPoolMXFP8）的注册路径与统一 radix 树默认化。