PR #50390 完整报告

vllm-project/vllm

[EPD] Remove duplicate image preprocessing in EPD and enable preprocess on GPU

合并时间：2026-08-06 06:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50390

执行摘要

- 一句话：EPD 消除重复图像预处理，grid 带外传递并启用 GPU 预处理
- 推荐动作：值得精读。核心看点：一是 EPD 场景下 " 信息该由计算方产出且只计算一次 " 的推导（grid 从 encoder 上报、代理不二次推导），二是用声明式 embedding_fields（values/metadata 角色）同时服务 producer 发布与 consumer 校验、消除两端契约漂移，三是 --mm-processor-device=auto 的复合门控（EC producer 非 consumer + torch_shm 传输）与 validate_mm_processor_device 的非 EPD 拒绝策略，四是 NO_REWRITE 诊断开关这种为可比性保留对照路径的做法。建议关注同类可扩展注入模式（如 PCPManager extensible）与这套声明的融合。

功能与动机

PR body 指出 EPD 部署存在两笔独立成本："The decode instance redoes work the encoder already did"——encoder 跑完整 HF 图像变换产出 embedding，decode 端还要对相同像素跑一遍相同变换，但真正需要的只是确定 placeholder 范围的 patch grid；同时 " The transform runs on CPU, and it is slow there"，2048² 图像 CPU 需要 11.46 ms 而 GPU 只要 4.32 ms。encoder 实例不分配 KV cache，"An encoder instance has the capacity to spare"，前端工作不与语言模型竞争。Issue 评论中 noooop 质疑在入口进程内初始化 GPU 的做法（"Preprocessing can interfere with the GPU computation, and most of the time it's gpu:0"），DarkLight1337 和 Isotr0py 均回应这正是利用 encoder 实例负载低的特性，最终方向被接受。

实现拆解

1. grid 元数据带外传递（Change 1）：在 vllm/distributed/ec_transfer/ec_connector/example_connector.py 中，ECExampleConnector 新增 request_finished 覆写，producer 侧把每个 request.mm_features 的 identifier（mm_hash）和经 _placeholder_metadata_fields 筛选的元数据（例如 image_grid_thw）打包进 ec_transfer_params["ec_items"] 返回给代理；_placeholder_metadata_fields 从 MultiModalDataParser.embedding_fields 读取，与 consumer 端 parser 的声明同源、不会漂移。在 examples/disaggregated/disaggregated_encoder/disagg_epd_proxy.py 中，代理新增 content_uuid（内容派生的缓存键，保证与未改写路径的哈希一致）、_b64_tensor（grid 序列化）和 rewrite_for_decode（把 image_url 条目改写为 {"type": "image_embeds", "image_embeds": {grid}, "uuid": ...}），并且明确规定 grid 只来自

encoder 上报、代理绝不二次推导。decode 侧 vllm/model_executor/models/qwen2_vl.py 等 parser 通过 MultiModalDataParser.embedding_field_sets 拿到 (required, optional) 字段集，在 EC consumer 上允许省略 embeds 但 metadata 字段仍必须存在，否则快速失 败。

2. GPU 预处理（Change 2）：vllm/config/multimodal.py 新增 MMProcessorDevice 类型与三个方法：fold_mm_processor_device 把 --mm-processor-device 便利 flag 折入 mm_processor_kwargs["device"]（状态不单独保存，显式 kwargs 优先）；get_mm_processor_device_type 用 torch.device 归一化以兼容 cuda:1、torch.device 等形式；validate_mm_processor_device 在非 encode-only 实例上拒绝加速器预处理并给出明确报错。vllm/config/vllm.py 的 __post_init__ 依次调用 _resolve_mm_embeds_from_ec_connector（仅 EC consumer 置 mm_embeds_from_ec_connector=True，其他场景保持 fail fast）、_resolve_mm_processor_device（auto 仅在 EC producer 非 consumer 且 mm_tensor_ipc == "torch_shm" 时解析为加速器，否则 CPU 并打日志）、_validate_mm_processor_device。vllm/multimodal/processing/context.py 的 _postprocess_output 只在 torch_shm 传输下保留设备张量，其他传输强制 .cpu()；vllm/multimodal/inputs.py 新增 _nested_tensors_are_cpu，reduce_data 在数据已位于设备时跳过 pin_memory（修复 cannot pin 'CUDABFloat16Type' 崩溃）。
3. 模型侧数据契约改造：vllm/multimodal/parse.py 引入 EmbeddingFieldRole（values/metadata）与 embedding_fields 类级声明，DictEmbeddingItems 支持 optional_fields 且全 optional 时抛错防止静默生成错误 prompt；vllm/model_executor/models/minicpmv.py、keye.py、keye_vl1_5.py、qwen2_vl.py、hunyuan_vision.py、llava_onevision2.py、colqwen3.py、colqwen3_5.py、opencua.py 等 9 个模型 parser 由硬编码 required_fields 改为声明式 embedding_fields + embedding_field_sets。
4. 配套测试与配置：vllm/config/ec_transfer.py 增加 is_encode_only 属性用于角色判断；vllm/engine/arg_utils.py 新增 CLI 参数 --mm-processor-device；tests/config/test_multimodal_config.py 新增 device 归一化、fold 优先级、auto 解析条件、非 EPD 拒绝、CPU 无门槛等 9 组测试，并守卫 VllmConfig 到校验函数的接线（"Startup must reach the check"）。代理侧还保留 NO_REWRITE 诊断开关，用于对照实验隔离改写路径的差异。

关键文件：

- examples/disaggregated/disaggregated_encoder/disagg_epd_proxy.py（模块 代理脚本；类别 source；类型 dependency-wiring；符号 content_uuid, _b64_tensor, rewrite_for_decode）：EPD 代理是 Change 1 的执行入口：新增 content_uuid/_b64_tensor/rewrite_for_decode，把图像条目改写为仅含 grid 元数据的引用，并收集 encoder 经 ec_transfer_params 上报的 mm_hash 与 grid；NO_REWRITE 开关维护对照路径。
- vllm/config/multimodal.py（模块 配置层；类别 source；类型 dependency-wiring；符号 fold_mm_processor_device, get_mm_processor_device_type, validate_mm_processor_device）：MultiModalConfig 新增 mm_embeds_from_ec_connector 派生字段、MMProcessorDevice 类型与

fold/get/validate 三个方法，是 GPU 预处理配置规则的所有者与校验集中地。

- vllm/config/vllm.py（模块 配置层；类别 source；类型 core-logic；符号 _resolve_mm_embeds_from_ec_connector, _resolve_mm_processor_device, _validate_mm_processor_device）：VllmConfig.__post_init__串起三个派生 / 校验步骤，把 EC 角色解析为 mm_embeds_from_ec_connector 与具体 processor device，是全部推导逻辑的总装点。
- vllm/multimodal/parse.py（模块 多模态层；类别 source；类型 core-logic；符号 placeholder_metadata_fields, embedding_field_sets, EmbeddingFieldRole）：引入 EmbeddingFieldRole 与 embedding_fields 声明式契约，DictEmbeddingItems 支持 optional_fields，是 EC consumer 省略 embeds 的解析层基础，也是 producer 端发布元数据的同源依据。
- vllm/distributed/ec_transfer/ec_connector/example_connector.py（模块 连接器；类别 source；类型 dependency-wiring；符号 _placeholder_metadata_fields, request_finished）：ECExampleConnector 的 request_finished 把 encoder 实际产出的 grid 元数据与 mm_hash 打包进 ec_transfer_params，是带外元数据通道的服务端实现。
- vllm/multimodal/processing/context.py（模块 多模态层；类别 source；类型 core-logic；符号 embeds_from_ec_connector）：InputProcessingContext._postprocess_output 按 mm_tensor_ipc 决定是否保留设备张量，embeds_from_ec_connector 属性把派生配置接到 parser 构造，是 GPU 预处理与 torch_shm 传输协同的关键。
- tests/config/test_multimodal_config.py（模块 配置测试；类别 test；类型 test-coverage；符号 test_mm_processor_device_type_normalizes, test_bad_mm_processor_device_rejected, test_accelerator_mm_processor_rejected_outside_encoder_instance, test_accelerator_mm_processor_allowed_on_encoder_instance）：新增 9 组测试覆盖 device 归一化、fold 优先级、auto 解析门控、非 EPD 拒绝与 CPU 无门槛，并守卫 VllmConfig 到校验函数的接线。
- vllm/engine/arg_utils.py（模块 引擎参数；类别 source；类型 core-logic；符号 mm_processor_device, create_model_config, add_cli_args）：新增 --mm-processor-device CLI 参数，是用户入口；choices 动态包含当前平台加速器名，并打包进入 ModelConfig 构造。
- vllm/multimodal/inputs.py（模块 多模态层；类别 source；类型 core-logic；符号 _nested_tensors_are_cpu）：新增 _nested_tensors_are_cpu，reduce_data 在批量数据已位于设备时跳过 pin_memory，修复设备张量 pin 崩溃，是 GPU 预处理落地的配套修复。
- vllm/config/ec_transfer.py（模块 配置层；类别 source；类型 core-logic；符号 is_encode_only）：新增 is_encode_only 属性，供 _resolve_mm_processor_device 与 validate_mm_processor_device 判断是否 EC producer 且非 consumer。
- vllm/model_executor/models/minicpmv.py（模块 模型解析；类别 source；类型 data-contract；符号 MiniCPMVMultiModalDataParser, MiniCPMVImageEmbeddingItems, MiniCPMVVideoEmbeddingItems）：首个完整迁移到 embedding_fields 声明式契约的模型 parser 之一，演示 values/metadata 角色如何在 MiniCPMV 图像 / 视频两类 embedding 上使用。

- vllm/model_executor/models/qwen2_vl.py（模块 模型解析；类别 source；类型 data-contract；符号 Qwen2VLMultiModalDataParser）：PR body 中明确改造的多模态 parser：Qwen2VLMultiModalDataParser 需接受只携带 grid 的 item，是 EC consumer 场景的直接受益者与验证者。
- vllm/model_executor/models/keye.py（模块 模型解析；类别 source；类型 data-contract）：同样迁移到 embedding_fields 契约的模型之一，验证契约覆盖更多模型类型。

关键符号：rewrite_for_decode, content_uuid, _b64_tensor, request_finished, _placeholder_metadata_fields, embedding_field_sets, placeholder_metadata_fields, fold_mm_processor_device, get_mm_processor_device_type, validate_mm_processor_device, _resolve_mm_embeds_from_ec_connector, _resolve_mm_processor_device, _validate_mm_processor_device, _nested_tensors_are_cpu, is_encode_only

关键源码片段

examples/disaggregated/disaggregated_encoder/disagg_epd_proxy.py

EPD 代理是 Change 1 的执行入口：新增 content_uuid/_b64_tensor/rewrite_for_decode，把图像条目改写为仅含 grid 元数据的引用，并收集 encoder 经 ec_transfer_params 上报的 mm_hash 与 grid；NO_REWRITE 开关维护对照路径。

```
def content_uuid(item: dict) -> str:
    """多模态条目的缓存键，由内容本身派生。

    必须基于内容而非请求派生：EC 缓存以该值为 key 索引，若改用请求 id，
    每次请求都会 miss，丢掉跨请求的媒体编码复用；而未改写路径按内容哈希，
    两者不对称会静默污染对照实验的结论。
    """
    url = (item.get("image_url") or item.get("audio_url") or {}).get("url") or ""
    payload = url or json.dumps(item, sort_keys=True)
    return hashlib.sha256(payload.encode()).hexdigest()


def _b64_tensor(values: list) -> str:
    """把 grid 元数据序列化为 base64 字符串，便于放进 JSON 请求体。"""
    import torch

    buf = io.BytesIO()
    grid = torch.tensor(values, dtype=torch.long)
    # 下游按条目堆叠，只保留扁平的 (t, h, w) 三个值
    torch.save(grid.reshape(-1)[:3], buf)
    return base64.b64encode(buf.getvalue()).decode()


def rewrite_for_decode(req_data: dict, item_meta: dict[int, dict]) -> dict:
    """把每个图像条目改写成仅含元数据的引用，再转发给 decode 实例。

    解码端不需要像素：encoder 已生成 embedding 并通过 EC 连接器按 uuid 发布，
```

```
    只发送 grid 即可让解码端确定 placeholder 占位范围，无需重跑图像变换。
    `item_meta` 是编码器上报的缓存键和它实际计算出的 grid，代理绝不在此
    二次推导 —— 二次推导可能与编码器结果不一致。
    """
    rewritten = 0
    idx = 0
    new_messages = []
    for msg in req_data.get("messages", []):
        content = msg.get("content")
        if not isinstance(content, list):
            new_messages.append(msg)
            continue
        new_content = []
        for item in content:
            if item.get("type") not in MM_TYPES:
                new_content.append(item)
                continue
            meta = dict(item_meta.get(idx) or {})
            idx += 1
            item_uuid = meta.pop("mm_hash", None)
            # 编码器上报了哪些键，就说明这是模型声明为确定占位尺寸所需的
            # 元数据，代理不需要知道键名
            metadata = {k: _b64_tensor(v) for k, v in meta.items()}
            if not metadata or not item_uuid:
                # 编码器没上报元数据（例如条目命中 processor 缓存），
                # 让解码端自己处理媒体，保证功能正确
                new_content.append(item)
                continue
            new_content.append(
                {
                    "type": "image_embeds",
                    "image_embeds": metadata,
                    "uuid": item_uuid,
                }
            )
            rewritten += 1
        new_messages.append({**msg, "content": new_content})

    if not rewritten:
        return req_data
    logger.info("Rewrote %d image item(s) as metadata references", rewritten)
    return {**req_data, "messages": new_messages}
```

vllm/config/multimodal.py

MultiModalConfig 新增 mm_embeds_from_ec_connector 派生字段、MMProcessorDevice 类型与 fold/get/validate 三个方法，是 GPU 预处理配置规则的所有者与校验集中地。

```
  @staticmethod
  def fold_mm_processor_device(
```

```
    mm_processor_kwargs: dict[str, Any] | None,
    mm_processor_device: MMProcessorDevice | None,
) -> dict[str, Any] | None:
    """把 `mm_processor_device` 便利 flag 折入 kwargs。

    flag 本身不保留状态：`mm_processor_kwargs["device"]` 是处理器运行
    位置的唯一表示，因此显式 device 总是优先，而 "auto" 保持未解析，
    留给持有 EC 角色信息的 `VllmConfig` 去决定。
    """
    if mm_processor_device in (None, "auto"):
        return mm_processor_kwargs
    if (mm_processor_kwargs or {}).get("device") is not None:
        return mm_processor_kwargs

    from vllm.platforms import current_platform

    # 除 "cpu" 外的任何显式值都表示加速器，因此程序化设置的 "cuda"
    # 在设备类型名为 "xpu" 的平台上仍然可用，无加速器时退化为 CPU
    device = (
        "cpu"
        if mm_processor_device == "cpu"
        else (current_platform.device_type or "cpu")
    )
    return {**(mm_processor_kwargs or {}), "device": device}


def validate_mm_processor_device(self, ec_config: ECTransferConfig | None) -> None:
    """校验本部署形态下 mm_processor_kwargs["device"] 是否合法。

    唯一执行该校验的位置，因此 CPU-only 平台也会先解析 device 再做早退，
    保证拼写错误永远在启动期暴露，而不是静默落到错误位置运行。
    """
    from vllm.platforms import current_platform

    device_type = self.get_mm_processor_device_type()
    accelerator = current_platform.device_type
    if device_type is None or accelerator in ("", "cpu"):
        return
    if device_type != accelerator:
        return

    if ec_config is None or not ec_config.is_encode_only:
        raise ValueError(
            "Cannot run the multi-modal processor on "
            f"{device_type!r}: this instance also runs the language model. "
            "The processor would share the device with the model's forward "
            "pass, and its allocations are outside the memory profiled for "
            "the KV cache -- risking OOM or a silently shrunken cache.\n"
            "Accelerator preprocessing is only supported on an encode-only "
```

```
            "instance of an encode/prefill/decode deployment (an EC producer "
            "that is not also a consumer).\n"
            'Use --mm-processor-device=cpu, or drop "device" from '
            "--mm-processor-kwargs."
        )
```

vllm/config/vllm.py

VllmConfig.__post_init__串起三个派生 / 校验步骤，把 EC 角色解析为 mm_embeds_from_ec_connector 与具体 processor device，是全部推导逻辑的总装点。

```
  def _resolve_mm_processor_device(self) -> None:
      """在 EC 角色已知后决定 `--mm-processor-device=auto` 的具体值。

      "auto" 的含义是：仅在处理器能独享加速器、且输出可免拷回主机时使用
      加速器 —— 即 encode-only 实例且张量传输支持设备张量。其余部署一律留在
      CPU。显式 device 已由 MultiModalConfig.fold_mm_processor_device 折入
      kwargs，这里不干预，只交给 _validate_mm_processor_device 校验。
      """
      model_config = self.model_config
      if model_config is None:
          return
      mm_config = model_config.multimodal_config
      if mm_config is None:
          return
      if mm_config.get_mm_processor_device_type() is not None:
          return

      from vllm.platforms import current_platform

      device_type = current_platform.device_type
      if device_type in ("", "cpu"):
          return

      ec_config = self.ec_transfer_config
      # EC producer 且非 consumer 的实例不跑 forward、不分配 KV cache，
      # 前端加速器工作可以独占设备
      if ec_config is None or not ec_config.is_encode_only:
          return

      if mm_config.mm_tensor_ipc != "torch_shm":
          # 其他传输序列化主机字节，输出会被拷回主机，
          # 该拷贝成本高于在设备上跑变换省下的时间
          logger.info_once(
              "EPD encoder instance: keeping the multi-modal processor on CPU "
              "because mm_tensor_ipc=%s cannot carry device tensors. Add "
              "--mm-tensor-ipc=torch_shm to run it on the accelerator.",
              mm_config.mm_tensor_ipc,
          )
          return
```

```
    mm_config.mm_processor_kwargs = {
        **(mm_config.mm_processor_kwargs or {}),
        "device": device_type,
    }
    logger.info_once(
        "EPD encoder instance: running the multi-modal processor on %s. "
        "Override with --mm-processor-device=cpu.",
        device_type,
    )
```

评论区精华

GPU 预处理是否值得（noooop 质疑，DarkLight1337/Isotr0py 支持）：noooop 说 "Preprocessing can interfere with the GPU computation, and most of the time it's gpu:0... I don't think we should initialize and use the GPU inside entrypoint"；DarkLight1337 回应 "The idea here is to run preprocessing on the Encoder instance's GPU which has lower workload than the Decode instance's GPU to begin with"，Isotr0py 表示同意（"we can still run GPU preprocessing on encoder instance for EPD deployment"）。结论：方向被接受，但通过 validate_mm_processor_device 硬性限制只有 encode-only 实例能用加速器。

mm_processor_device 是否冗余（DarkLight1337）："I think mm_processor_device is redundant, if the same thing could be achieved by setting --mm-processor-kwargs directly. If it's just for user convenience then we should make this init-only and store the state exclusively via mm_processor_kwargs field"。结论：采纳，flag 仅作便利入口，状态统一存 mm_processor_kwargs["device"]。

校验位置（Isotr0py）："Let's validate it in MultimodalConfig instead"，随后第 2 轮又建议 "Let's move the resolve to vllm_config or multimodal_config's __post_init__"。结论：采纳，最终 validate 归属 MultiModalConfig，resolve 移入 VllmConfig.__post_init__。

避免硬编码 *_grid_thw（DarkLight1337）："I think we should think about how to extend this for models that require keys other than *_grid_thw to avoid hardcoding the attribute names"；Isotr0py 进一步建议用 dict 的 value/key 表达 metadata 角色（"we can present metadata/out_of_band fields as dict's values/keys"）。结论：采纳，演进为 embedding_fields 声明式契约，producer 与 consumer 同源。

非 EPD 部署必须拒绝 GPU 预处理（Isotr0py）："User may try to enable GPU preprocessing with mm_processor_device="gpu" / mm_processor_kwargs={"device": "cuda"}, even if they aren't launching EPD deployment"。结论：采纳，启动时 raise 并给出 KV cache 越界分配警告。

命名从 out_of_band 改为 embeds_from_ec_connector：Isotr0py 追问 "can we assume all values are from out of band..."，gty111 回应 "Now, we remove allow_out_of_band since it is vague and change it to embeds_from_ec_connector"，让语义精确到 EC connector 这一唯一来源。

- GPU 预处理是否值得做、是否该在 entrypoint 初始化 GPU (design): 方向被接受：GPU 预处理限定在 encode-only 实例，且通过 validate_mm_processor_device 从配置层硬性拒绝其他部署形态。
- mm_processor_device 与 mm_processor_kwargs 是否重复 (design): 采纳：flag 仅作 CLI 便利入口，状态统一存于 mm_processor_kwargs["device"]，由 fold_mm_processor_device 折入且显式 kwargs 优先。
- 校验与解析逻辑应放哪里 (design): 采纳：validate 归属 MultiModalConfig，resolve 移入 VllmConfig.post_init，EngineArgs 只保留 CLI 透传。
- 硬编码 *_grid_thw 的可扩展性问题 (design): 采纳：演进为 embedding_fields 声明式契约（EmbeddingFieldRole），producer 发布与 consumer 校验同源。
- 非 EPD 部署应拒绝 GPU 预处理 (correctness): 采纳：validate_mm_processor_device 在非 encode-only 实例上启动期 raise，并给出与 KV cache 共享显存的风险说明。
- out_of_band 命名语义模糊 (design): 采纳：命名精化为 embeds_from_ec_connector，明确 embedding 的带外来源就是 EC connector。
- _nested_tensors_are_cpu 实现简化 (style): 采纳简化实现。

风险与影响

- 风险：数据契约变更面广：vllm/multimodal/parse.py 的 DictEmbeddingItems 签名变化（required_fields/optional_fields/embedding_field_sets）牵动 9 个模型 parser，任何模型 embedding_fields 声明与实际 processor 输出不一致，都会导致解析期报错或（更危险的）占位范围被静默算错；DictEmbeddingItems 已补全 optional 校验兜底。

派生配置的误判风险：mm_embeds_from_ec_connector 由 VllmConfig._resolve_mm_embeds_from_ec_connector 无条件覆写，若 ec_transfer_config.is_ec_consumer 判定异常（例如 connector 配置错误），非 EC 场景可能被放行缺失 embeds 的请求；反方向，EC consumer 场景若该标志未生效，用户请求会直接在前端报缺字段错误。

GPU 预处理的内存越界风险：虽然仅 encode-only 实例可用，但 HF processor 跑在 API-server 进程内，其显存分配仍不在 engine 的 KV cache profiling 范围内；若 encoder 与 decode 共享同一物理 GPU（测试中就是 encoder GPU0 + decode GPU1），快速傅里叶等前端算子仍可能与模型计算竞争。

传输方式相关回归：_postprocess_output 只在 mm_tensor_ipc == "torch_shm" 时保留设备张量，direct_rpc 下强制 .cpu()；如果未来新增可承载设备张量的传输类型而忘记同步该判定，会引入不必要的拷贝或把设备张量送到不允许的主机序列化路径。reduce_data 的 pin_memory 跳过依赖 _nested_tensors_are_cpu 的全量叶子遍历，批体积大时有少量开销。

代理与 encoder 版本耦合：disagg_epd_proxy.py 依赖 encoder 通过 ec_transfer_params 返回 ec_items，若 encoder 侧版本未升级，rewrite_for_decode 会 fallback 让 decoder 自处理媒体（功能正确但性能回退），且 content_uuid 与未改写路径的哈希一致性是缓存命中的前提，NO_REWRITE 开关旁路时需保持该一致性。

- 影响：用户 / 部署侧：EPD 部署吞吐提升 1.18–1.31×（Qwen3.5-35B-A3B bf16，2048² 图像），TTFT 降 17–34%，TPOT 在 c=32 时由 25.7 ms 降至 22.9 ms；decode 实例每

个请求少收 3.36 MB base64 像素并免去重跑变换。新增 --mm-processor-device CLI 选项（默认 auto），对所有未启用 EPD 的用户为纯透明变更。

代码库结构：多模态解析层新增 embedding_fields 声明式契约，未来新模型接入 EPD out-of-band 路径只需声明 metadata 字段角色；EC connector 的 request_finished 钩子成为 producer 上报元数据的标准通道。

团队协作：该 PR 经过 27 条 review 评论、22 个 commit 的演进（硬编码 grid → 声明式字段、out_of_band → embeds_from_ec_connector、arg_utils 校验 → MultimodalConfig 校验），确立了 " 派生配置不暴露 CLI、契约声明两端复用、校验位置集中 " 的代码组织原则，对后续 EPD 功能开发有示范作用。

- 风险标记：核心路径变更，数据契约变更，跨模块联动，新增配置选项，多模型解析器联动

关联脉络

- PR #44956 [KV Connector][Mooncake] Add store group semantics: 同为 EC/kv-transfer 连接器基础设施演进，扩展了连接器元数据与对象组织语义，与本 PR 的 ec_transfer_params 元数据通道属于同一能力线。
- PR #50507 [KV Offloading] Support partial-tail prefix reuse with fine-grained prefix matching: KV 连接器 / 卸载方向的缓存复用优化，与本 PR 通过 EC connector 复用 encoder 计算结果（避免重复变换）同属消除跨实例重复计算的主题。
- PR #50358 [Bugfix] Fail fast with a clear error when CPU offload region exceeds available space: 同为 v1 特性中的快速失败策略实践：本 PR 在非 EC 场景缺失 embeds 时也强调前端快速失败，二者设计取向一致。
- PR #50066 [Refactor][PCP] Make PCPManager construction extensible: PCPManager 可注入子类的可扩展构造模式，与本 PR embedding_fields 声明式契约、MultiModalConfig 持有规则的设计思路相近，可作后续统一演进的参考。