PR #33936 完整报告

sgl-project/sglang

feat(vlm): auto-select CUDA VMM on multi-node MNNVL

合并时间：2026-08-08 16:00

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33936

执行摘要

- 一句话：多节点 MNNVL 自动启用 CUDA VMM 多模态传输
- 推荐动作：值得精读。重点看三个设计决策：一是 server_args._handle_multimodal_feature_transport 的分层自动选择策略（单节点 cuda_ipc / 多节点 MNNVL cuda_vmm / cpu 兜底），是 " 安全默认值自动选择 " 的范例；二是模型类属性 supports_cuda_vmm_feature_transport 作为 opt-in 契约的模式，扩展成本极低；三是 _resolve_transport_consumer_count 的钳制逻辑，解决了 DP 注意力下全 TP 确认的错误释放问题。建议阅读时结合 #33899 的传输实现，并关注后续多节点实测 PR。

功能与动机

多节点多模态部署（GB200/GB300 MNNVL）此前没有 GPU 直传路径：CUDA IPC handle 仅限单节点内，跨节点特征只能走 host payload，ViT 特征需要先拷到 CPU 再上 GPU。PR body 明确说明优化目标是去除 host payload 路径："CUDA VMM exports a POSIX FD on one node and a CUDA FABRIC handle on multi-node MNNVL. Consumers still stage remote HBM into local HBM before ViT; the optimization removes the host payload path, not that local staging copy." 同时希望免去手工指定："Auto-select cuda_vmm for validated multi-node GB200/GB300 MNNVL deployments when an IMEX channel is mounted"，并保留 cpu 为显式 opt-out 与安全回退。

实现拆解

1. 传输参数解析与自动选择重写（python/sglang/srt/server_args.py）：更新 mm_feature_transport 的参数说明，明确三档自动策略；_handle_multimodal_feature_transport 的自动分支从 " 单节点 cuda_ipc / 其他 cpu" 扩展为：单节点 CUDA 继续选 cuda_ipc；多节点且 is_mnnvl_fabric_device() 为真、/dev/nvidia-caps-imex-channels/channel0 存在、模型 opt-in 时选 cuda_vmm；其余情况回退 cpu，并区分 " 模型未 opt-in" 与 "MNNVL 已检测但无 IMEX channel" 两类日志。显式 cuda_vmm 的校验（NVIDIA CUDA、pp_size == 1、非 Rust server）不变，新增启动日志：handle 类型（nnodes > 1 时为 CUDA FABRIC，否则 POSIX FD）、HBM 预算、base GPU、worker 数与 CPU fallback 行为。
2. 模型 opt-in 契约（python/sglang/srt/model_loader/utils.py + models/kimi_k25.py、models/kimi_k3.py、models/qwen3_vl.py）：新增 supports_cuda_vmm_feature_transport(model_config)，解析出模型类后读取类属性

supports_cuda_vmm_feature_transport；三个模型类各加一行 supports_cuda_vmm_feature_transport = True。自动选择门控据此按模型白名单收窄，未标记模型在多节点自动回退 cpu。

3. 延迟重构路径与消费确认钳制（python/sglang/srt/managers/schedule_batch.py）：MultimodalDataItem.reconstruct 与 acknowledge_deferred_cuda_ipc_feature 经新增的 _resolve_transport_consumer_count 将传入的 consumer count 钳制到 proxy 声明的 total_consumer_count/consumer_count。原因是 CUDA VMM 的 proxy 在 DP 注意力下消费者集合可能小于全 TP 秩数，按全 TP 数确认会造成错误释放；precomputed_embeddings 与 model_specific_data 中的代理仍走即时物化，不进入延迟路径。
4. 处理器开关泛化（multimodal/processors/kimi_k25.py、multimodal/processors/kimi_k3.py）：延迟重构的标记条件从 self.use_cuda_ipc / getattr(self, "use_cuda_ipc", False) 改为 self.keep_mm_features_on_device，使 cuda_vmm 传输也走 deferred encoder-owner 路径；kimi_k25 保留 self.server_args.mm_enable_dp_encoder 的附加门控。
5. 测试与文档配套：test/registered/unit/server_args/test_server_args.py 新增 6 个策略用例（支持 / 不支持模型的多节点 MNNVL 自动选择、无 IMEX channel、非 MNNVL 多节点、legacy keep_mm_feature_on_device 与显式 cuda_vmm 冲突、显式 cuda_vmm 共享 HBM 预算并兼容 legacy 环境变量）；test/registered/unit/models/test_kimi_k25.py 新增 proxy consumer count 钳制与 cache-hit 确认行为测试；docs/cookbook/autoregressive/Moonshotai/Kimi-K3.mdx增加自动策略与HBM权衡说明。

关键文件：

- python/sglang/srt/server_args.py（模块 参数解析；类别 source；类型 dependency-wiring；符号 _handle_multimodal_feature_transport）：核心自动选择策略所在：_handle_multimodal_feature_transport 扩展多节点 MNNVL 分支，显式 cuda_vmm 新增 handle 类型与预算日志，参数帮助文案同步更新。
- python/sglang/srt/managers/schedule_batch.py（模块 调度器；类别 source；类型 core-logic；符号 _resolve_transport_consumer_count, reconstruct, acknowledge_deferred_cuda_ipc_feature）：新增 _resolve_transport_consumer_count，将 reconstruct 与延迟确认的 consumer count 钳制到 proxy 实际消费者集合，是 DP 注意力下 CUDA VMM 安全释放的关键。
- python/sglang/srt/model_loader/utils.py（模块 模型加载；类别 source；类型 data-contract；符号 supports_cuda_vmm_feature_transport）：新增 supports_cuda_vmm_feature_transport 契约入口，自动选择门控据此判断模型类是否 opt-in，是模型准入的单一事实来源。
- test/registered/unit/server_args/test_server_args.py（模块 参数测试；类别 test；类型 test-coverage；符号 test_legacy_keep_flag_rejects_explicit_cuda_vmm, test_default_transport_is_cuda_vmm_for_supported_multinode_mnnvl, test_default_transport_is_cpu_for_unsupported_multinode_model, test_default_transport_is_cpu_without_imex_channel）：6 个新测试用例覆盖自动选择策略的每一路分支（支持 / 不支持模型、缺 IMEX、非 MNNVL、legacy 冲突、共享预算），

是本次默认行为变更的主要保障。

- test/registered/unit/models/test_kimi_k25.py（模块 模型测试；类别 test；类型 test-coverage；符号 test_kimi_lazy_vmm_feature_uses_proxy_consumer_count, test_kimi_lazy_vmm_cache_hit_uses_proxy_consumer_count）：验证 CUDA VMM proxy 下 reconstruct 与 cache-hit 确认均使用 proxy 自身 consumer_count，防止回归到全 TP 确认。
- python/sglang/srt/models/kimi_k25.py（模块 Kimi 模型；类别 source；类型 data-contract；符号 supports_cuda_vmm_feature_transport）：Kimi-K2.5/K2.7 模型类 opt-in 标记，接入 CUDA VMM 自动选择。
- python/sglang/srt/models/kimi_k3.py（模块 Kimi 模型；类别 source；类型 data-contract；符号 supports_cuda_vmm_feature_transport）：Kimi-K3 模型类 opt-in 标记，与 K2.5 系列一同接入 CUDA VMM。
- python/sglang/srt/models/qwen3_vl.py（模块 Qwen 模型；类别 source；类型 data-contract；符号 supports_cuda_vmm_feature_transport）：Qwen3-VL 系列模型类 opt-in 标记，扩展 CUDA VMM 覆盖范围。
- python/sglang/srt/multimodal/processors/kimi_k25.py（模块 MM 处理器；类别 source；类型 core-logic）：延迟重构开关从 use_cuda_ipc 泛化为 keep_mm_features_on_device，使 CUDA VMM 也走 deferred encoder-owner 路径。
- python/sglang/srt/multimodal/processors/kimi_k3.py（模块 MM 处理器；类别 source；类型 core-logic）：同上，K3 处理器开关泛化；原先用 getattr 防御，现直接访问实例属性。
- docs/cookbook/autoregressive/Moonshotai/Kimi-K3.mdx（模块 文档；类别 docs；类型 documentation）：Kimi-K3 cookbook 补充自动策略说明与 HBM 预算权衡，帮助用户理解默认值变化与显式 opt-out 方式。

关键符号：_handle_multimodal_feature_transport, supports_cuda_vmm_feature_transport, _resolve_transport_consumer_count, MultimodalDataItem.reconstruct, MultimodalDataItem.acknowledge_deferred_cuda_ipc_feature

关键源码片段

python/sglang/srt/server_args.py

核心自动选择策略所在：_handle_multimodal_feature_transport 扩展多节点 MNNVL 分支，显式 cuda_vmm 新增 handle 类型与预算日志，参数帮助文案同步更新。

```
# _handle_multimodal_feature_transport 的自动选择核心分支（head 版本）
# 前置：requested_transport 已处理 legacy 环境变量与 --keep-mm-feature-on-device 映射
elif (
    self.get_model_config().is_multimodal
    and is_cuda()
    and self.disaggregation_mode == "null"
):
    # 单节点 CUDA 部署继续自动使用 cuda_ipc 有界池；
    # 多节点自动切 cuda_vmm 需同时满足：GB200/GB300 MNNVL 设备、
```

```
    # IMEX channel 已挂载、且模型类显式 opt-in（supports_cuda_vmm_feature_transport）
    if self.nnodes == 1:
        requested_transport = "cuda_ipc"
        logger.info(
            "Multimodal feature transport auto-resolved to cuda_ipc "
            "(single-node CUDA). Pass --mm-feature-transport=cpu to opt out."
        )
    elif is_mnnvl_fabric_device() and os.path.exists(
        "/dev/nvidia-caps-imex-channels/channel0"
    ):
        from sglang.srt.model_loader.utils import (
            supports_cuda_vmm_feature_transport,
        )

        if supports_cuda_vmm_feature_transport(self.get_model_config()):
            requested_transport = "cuda_vmm"
            logger.info(
                "Multimodal feature transport auto-resolved to cuda_vmm "
                "(multi-node GB200/GB300 MNNVL). Pass "
                "--mm-feature-transport=cpu to opt out."
            )
        else:
            # 模型未 opt-in：安全回退 cpu，不做 GPU 直传
            requested_transport = "cpu"
            logger.info(
                "Multimodal feature transport auto-resolved to cpu: "
                "the model has not opted into CUDA VMM transport."
            )
    else:
        requested_transport = "cpu"
        if is_mnnvl_fabric_device():
            # MNNVL 设备存在但无 IMEX channel，给出可操作的配置提示
            logger.info(
                "Multimodal feature transport auto-resolved to cpu: "
                "GB200/GB300 was detected but no IMEX channel is mounted. "
                "Configure the MNNVL compute domain or pass "
                "--mm-feature-transport=cuda_vmm after doing so."
            )

# 显式 cuda_vmm 的校验（CUDA/pp_size/Rust server）仍执行；
# 校验通过后新增启动日志：多节点用 CUDA FABRIC handle，单节点用 POSIX FD
if requested_transport == "cuda_vmm":
    pool_budget_mb = envs.SGLANG_MM_FEATURE_CACHE_MB.get()
    handle_kind = "CUDA FABRIC" if self.nnodes > 1 else "POSIX FD"
    logger.info(
        "Using CUDA VMM for multimodal features with %s sharing: "
        "reserving up to %d MiB on base GPU %d across %d tokenizer "
        "worker(s). This reduces KV cache headroom; a full pool falls "
        "back to inline CPU transport.",
```

```
            handle_kind,
            pool_budget_mb,
            self.base_gpu_id,
            self.tokenizer_worker_num,
        )
```

python/sglang/srt/managers/schedule_batch.py

新增 _resolve_transport_consumer_count，将 reconstruct 与延迟确认的 consumer count 钳制到 proxy 实际消费者集合，是 DP 注意力下 CUDA VMM 安全释放的关键。

```
    def reconstruct(self, target_device: int, ipc_consumer_count: int = 1):
        """在 target_device 上就地物化 CUDA IPC/VMM 代理张量"""
        if isinstance(self.feature, CudaIpcTensorTransportProxy):
            # DP 注意力下 image 只分发给实际消费者，consumer 集合可能小于全 TP 秩数；
            # 先钳制再物化，避免按全 TP 数重建导致资源语义错误
            consumer_count = self._resolve_transport_consumer_count(
                self.feature, ipc_consumer_count
            )
            if consumer_count == 1:
                self.feature = self.feature.reconstruct_on_target_device(target_device)
            else:
                self.feature = self.feature.reconstruct_on_target_device(
                    target_device, consumer_count=consumer_count
                )
        # precomputed_embeddings 与 model_specific_data 中的代理仍即时物化，不进延迟路径
        if isinstance(self.precomputed_embeddings, CudaIpcTensorTransportProxy):
            self.precomputed_embeddings = (
                self.precomputed_embeddings.reconstruct_on_target_device(target_device)
            )
        # model_specific_data 内代理同理逐项重建 ...

    def acknowledge_deferred_cuda_ipc_feature(self, consumer_count: int = 1):
        """embedding-cache 命中跳过 ViT 时，释放延迟的 GPU 传输特征"""
        if isinstance(self.feature, CudaIpcTensorTransportProxy):
            # cache 命中场景同样按 proxy 实际消费者数确认，防止超额释放
            consumer_count = self._resolve_transport_consumer_count(
                self.feature, consumer_count
            )
            self.feature.acknowledge_consumption(consumer_count)

    @staticmethod
    def _resolve_transport_consumer_count(proxy, requested_count: int) -> int:
        """把组确认数钳制到 proxy 的实际消费者集合"""
        # 优先取 proxy 声明的 total_consumer_count，其次 consumer_count；
        # 两者都没有时退回请求值，保持旧有 IPC 行为不变
        proxy_count = getattr(
            proxy,
            "total_consumer_count",
            getattr(proxy, "consumer_count", requested_count),
```

```
        )
        return min(requested_count, proxy_count)
```

python/sglang/srt/model_loader/utils.py

新增 supports_cuda_vmm_feature_transport 契约入口，自动选择门控据此判断模型类是否 opt-in，是模型准入的单一事实来源。

```
def supports_cuda_vmm_feature_transport(model_config: ModelConfig) -> bool:
    # 模型 opt-in 契约：解析出最终模型类后读取类属性；
    # 未标记的模型在 server_args 自动选择时回退 cpu 传输
    model_cls, _ = get_model_architecture(model_config)
    return bool(getattr(model_cls, "supports_cuda_vmm_feature_transport", False))


class KimiK25ForConditionalGeneration(nn.Module):
    # 类属性即 opt-in 标记：置 True 后多节点 MNNVL 自动选择才会启用 cuda_vmm
    supports_cuda_vmm_feature_transport = True
```

评论区精华

该 PR 没有任何 reviewer 评论（review comments 为 0），两条 issue 评论分别为 CI 重跑命令 /tag-and-rerun-ci extra 和 mintlify 文档预览 bot 通知，均无技术内容。真正的 " 讨论 " 体现在作者在 PR body 中自述的验证边界：

> "Strict multi-node CUDA VMM A/B is externally capacity-blocked: the 2026-08-08 rx xpus inventory reported 0 available GPUs and 0 free nodes on both GB300 clusters and both B300 clusters. A single-node H200/B200 run cannot validate the multi-node CUDA FABRIC handle or the MNNVL auto-selection gate."
>
> "Earlier prototype-FABRIC numbers were removed because they do not measure the CUDA VMM implementation now in this PR."
>
> "The optimization removes the host payload path, not that local staging copy."

作者明确承认多节点分支未实测，并主动剔除了与当前实现不对应的早期 prototype 数据，处理方式值得肯定。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 多节点验证空白：GB300/B300 集群在 PR 提交时无可用资源，自动选择门控（is_mnnvl_fabric_device + IMEX 探测）与 CUDA FABRIC handle 均无真实环境数据；显式 cuda_vmm 在单节点 H200 冒烟测试中只覆盖了 POSIX FD 分支。这部分是最大不确定性。
1. 默认行为变更：满足 MNNVL + IMEX + 模型 opt-in 的多节点部署会在无参数指定时自动切换到 cuda_vmm。若 FABRIC 栈未就绪，显式选择会直接报错而非回退；自动选择路径虽以 cpu 兜底，但依赖对 is_mnnvl_fabric_device() 与设备路径探测的正确性，驱动 / 容

器挂载差异可能导致误判。

2. 处理器属性依赖变化（multimodal/processors/kimi_k25.py、kimi_k3.py）：从带 getattr 防御的 use_cuda_ipc 改为直接访问 self.keep_mm_features_on_device，若某个构造路径未初始化该属性会抛 AttributeError（测试中已补 processor.mm_feature_transport = "cpu" fixture，说明该属性由构造期注入，需确认 base_processor 统一初始化）。
3. consumer 钳制的双向约束：_resolve_transport_consumer_count 在 proxy 无 total_consumer_count/consumer_count 属性时返回 requested_count，旧行为兼容；但若 proxy 声明的消费者数小于实际消费者，会出现欠确认（引用计数残留），该契约依赖 proxy 实现正确声明。
4. HBM 预算共享：cuda_vmm 与 cuda_ipc 共享 SGLANG_MM_FEATURE_CACHE_MB（默认 1 GiB），多节点 tokenizer worker 增多时每 worker 分得更少，池满后逐 tensor 回退 inline CPU 会抬高请求延迟。 - 影响：用户 / 部署：多节点 GB200/GB300 MNNVL 上运行 Kimi-K2.5/K2.7、Kimi-K3、Qwen3-VL 系列时，多模态特征传输自动从 host 拷贝切换到 GPU 直传（CUDA FABRIC handle），降低跨节点多模态请求延迟；cpu 兜底保证未 opt-in 模型与缺失 IMEX 的环境保持原有行为。系统：server_args 的自动选择策略、schedule_batch 的物化与确认路径、3 个模型类的 opt-in 契约、2 个处理器的开关语义均被修改，涉及多模态请求的主路径；后续新模型若要接入只需加一个类属性。团队：多节点 FABRIC 实测被外部容量阻塞，验证工作会顺延到后续 PR，需要持续跟踪。 - 风险标记：多节点 FABRIC 未实测，自动选择改变默认传输，处理器属性访问无兜底，共享 HBM 预算受限

关联脉络

- PR #33887 config: retire ServerArgs.derive; per-runner values are constructor arguments: 同一时期对 server_args.py 参数解析体系的系列重构，本 PR 的自动选择逻辑建立在 " 参数在构造期解析 " 的新模型之上；分支多次 merge main 时与 server_args.py 冲突合入。
- PR #33888 config: delete the dead get_server_args() bindings across the repo: 同样触及 schedule_batch.py 等核心文件的配置清理，与本次传输参数绑定改动处于同一演进主线。