

PR #39347 完整报告

sgl-project/sglang

Allow CUDA VMM feature transport with the Rust frontend

合并时间: 2026-09-15 08:09

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39347>

执行摘要

- 一句话: Rust 前端下放开 CUDA VMM 多模态传输限制
- 推荐动作: 改动小而清晰, 适合快速浏览。值得关注的是测试从拒绝到接受的转向, 以及后续 Rust 前端实际运行 CUDA VMM 的验证情况; 若关心 Rust 前端多模态路径, 可精读该变更并跟进后续运行时测试。

功能与动机

PR body 明确说明要 Remove the blanket rejection of `--mm-feature-transport=cuda_vmm` when `SGLANG_RUST_SERVER=1`, 即此前 Rust 前端会无差别拒绝该配置, 而现在需要放开限制, 让 Rust 前端同样可以使用 CUDA VMM 传输多模态特征。

实现拆解

1. 修改 `python/sglang/srt/arg_groups/serving_hook.py` 中 `handle_multimodal_feature_transport` 函数, 删除 `if envs.SGLANG_RUST_SERVER.get(): raise ValueError(...)` 分支, 使 `cuda_vmm` 传输不再被 Rust 前端环境变量拒之门外; NVIDIA CUDA 平台检查和 `pp_size == 1` 检查保持不变。
2. 更新 `test/registered/unit/server_args/test_server_args.py`: 将原 `test_cuda_vmm_rejects_rust_server` 重命名为 `test_cuda_vmm_allows_rust_server`, 移除 `assertRaisesRegex`, 改用 `patch.dict(os.environ, {}, clear=False)` 隔离环境变量后调用 `handle_multimodal_feature_transport`, 并断言 `resolution_result` 仍为 `cuda_vmm`, 验证选择成功。
3. 保留 `test_cuda_vmm_rejects_non_nvidia_platforms` 与 `test_cuda_vmm_rejects_pipeline_parallelism`, 确保 CUDA 与 PP 限制仍被覆盖。测试运行 18 个用例全部通过, 新回归在修复前会失败, 符合正向回归预期。

关键文件:

- `python/sglang/srt/arg_groups/serving_hook.py` (模块 参数校验; 类别 source; 类型 core-logic; 符号 `handle_multimodal_feature_transport`): 移除 Rust 前端对 `cuda_vmm` 的拒绝逻辑, 是本次行为变更的核心。
- `test/registered/unit/server_args/test_server_args.py` (模块 单元测试; 类别 test; 类型 test-coverage; 符号 `test_cuda_vmm_allows_rust_server`): 将拒绝 Rust 前端的测试改

为接受测试，验证放开后的行为。

关键符号：handle_multimodal_feature_transport

关键源码片段

[python/sglang/srt/arg_groups/serving_hook.py](#)

移除 Rust 前端对 cuda_vmm 的拒绝逻辑，是本次行为变更的核心。

```
def handle_multimodal_feature_transport(server_args: Any) -> None:
    # ... 前面处理 MNNVL、legacy IPC、encoder_only 等逻辑（省略）
    if requested_transport == "cuda_vmm":
        # 强制要求 NVIDIA CUDA 平台，否则直接报错
        if not get_platform().is_cuda:
            raise ValueError("--mm-feature-transport=cuda_vmm requires NVIDIA CUDA.")
        # 尚不支持流水线并行（pipeline parallelism），pp_size 必须为 1
        if cfg.pp_size != 1:
            raise ValueError(
                "--mm-feature-transport=cuda_vmm does not support pipeline parallelism."
            )
        # 此处原先在 SGLANG_RUST_SERVER=1 时拒绝，本次改动移除了该限制，
        # 允许 Rust 前端使用 CUDA VMM 传输多模态特征，后续预留与日志逻辑不变
        pool_budget_mb = envs.SGLANG_MM_FEATURE_CACHE_MB.get()
        handle_kind = "CUDA FABRIC" if cfg.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,
            cfg.base_gpu_id,
            cfg.tokenizer_worker_num,
        )
    # ... 后续 cuda_ipc 等分支（省略）
```

[test/registered/unit/server_args/test_server_args.py](#)

将拒绝 Rust 前端的测试改为接受测试，验证放开后的行为。

```
@override_platform(is_cuda=True)
def test_cuda_vmm_allows_rust_server(self):
    # 构造一个指定 cuda_vmm 的 server args
    server_args = ServerArgs(model_path="dummy", mm_feature_transport="cuda_vmm")

    # 隔离环境变量，并模拟 Rust 前端启用
    with (
        patch.dict(os.environ, {}, clear=False),
        envs.SGLANG_RUST_SERVER.override(True),
    ):
        # 不应再抛错，正常完成配置解析
```

```
handle_multimodal_feature_transport(server_args)

# Rust 前端下传输方式应保持 cuda_vmm
self.assertEqual(
    resolution_result(server_args, "mm_feature_transport"), "cuda_vmm"
)
```

评论区精华

该 PR 没有 review 评论；作者通过 /tag-and-rerun-ci 触发 CI 重跑。CI 状态显示 Extra 与 AMD ROCm 10 两个 job 为失败，但缺少具体失败日志，无法确认是否与本 PR 相关，仅作参考。

- 暂无高价值评论线程

风险与影响

- 风险：移除显式拒绝后，若 Rust 前端在实际运行时仍未完全支持 CUDA VMM，错误可能在启动后暴露而非配置期清晰报错。PR body 声明未进行 GPU serving 验证，仅 CPU 配置测试，因此存在实际运行兼容性风险。测试仅覆盖参数解析，未覆盖真实传输路径，回归覆盖有限。
- 影响：影响范围限于使用 Rust 前端并期望启用 CUDA VMM 多模态传输的用户，扩大了可选传输方式；现有 Python 前端行为完全不变。该变更仅删除一个校验分支，对系统其余模块无连带影响，整体风险较低。
- 风险标记：GPU 实测缺失，Rust 前端兼容性未验证

关联脉络

- 暂无明显关联 PR