

PR #39677 完整报告

sgl-project/sglang

dsv4.1: Rust extension modules for image preprocessing, KV pool names, and PD bootstrap

合并时间: 2026-09-16 09:11

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39677>

执行摘要

本 PR 是 DeepSeek-V4.1 支持的 Rust 侧前置落地 (+197/-0, 5 个文件, 纯新增), 包含三块互不耦合的能力: `sglang-mm` 新增 `dsv41` 图像预处理模块 (PIL bicubic resize + 居中 padding + CHW patchify + bf16 量化)、`sglang-radix-tree` 的 `PoolName` 补齐 `deepseek_v4_c1/c2` 系列 KV 池名并同步 Python 字符串映射、`sglang-server` 的 PD bootstrap 路由新增可选 `dsv41_spec_layout` 并在跨 prefill rank 布局不一致时返回 400。作者明确说明把 Rust 侧单拆是为了保住预编译扩展的缓存 key (key 会哈希 `rust/**`)。改动对既有路径零回归面, 价值集中在 "为后续 Python 侧改动铺路" 这一节奏控制上。

功能与动机

PR body 的第一句就是动机:

"Lands the Rust side of DeepSeek-V4.1 support ahead of the Python changes."

拆分的工程理由也写得很清楚:

"Landing this separately keeps the prebuilt-Rust-extension cache key stable for the Python-side changes that follow: the key hashes `rust/**`, so any PR carrying these files misses the cache and recompiles the extensions on every push."

要解决三个具体问题:

1. V4.1 图像预处理需要 Rust 加速: 新增 `dsv41::resize_patchify`, 对齐 PIL bicubic 语义并直接输出 bf16 bits。
2. HiCache 需要能识别 V4.1 的新 KV 池: `deepseek_v4_c1/c2` 及其 `indexer / indexer_scale` 变体此前不存在于 `PoolName` 枚举中。
3. PD 拆分需要跨 rank 校验 DSpark 投机布局: 新增 `dsv41_spec_layout`, 让布局冲突在 prefill 注册阶段就暴露为 400, 而非留到运行期。

验证方式为 `cargo test -p sglang-server --lib disaggregation` 5/5 通过, 含既有 `route_contract_matches_python_client`。

实现拆解

1) `sglang-mm` 新增 `dsv41` 图像预处理 (核心)

`rust/sglang-mm/src/dsv41/mod.rs` 定义了 `ImagePlan` (`out_h/out_w/resize_h/resize_w/top/left/patch_size`) 与 `resize_patchify(rgb, h, w, plan)`

-> Result<Vec<u16>, String>, 执行顺序为:

- 全量几何合法性校验, 一次性挡住除零、画布不可整分、resize 超出画布、padding 越界与输入长度不匹配 (checked_mul 防溢出);
- 经 crate::common::resize::resize_rgb 走 Resample::Pil(Filter::Bicubic);
- 用 256 项 bf16 LUT 把 (v/255-0.5)/0.5 查表写出, 内层循环无浮点运算;
- 用 par::for_chunks_mut 按 3*ps*ps 分块并行, patch 间无重叠所以天然安全; 越界像素填 127 (归一化后的 0)。

Python 侧通过 `#[cfg(feature = "python")] mod python` 暴露同名函数, 要求 HWC 且连续, 用 `py.detach` 释放 GIL。

2) sglang-mm 接线: `rust/sglang-mm/src/lib.rs` 加 `pub mod dsv41`; 与 `dsv41::register(m)?`。需要注意该 crate 的双构建形态——`default-features = false` 时是纯 Rust rlib 且不含 pyo3, `register` 只在 `python` feature 下存在。

3) sglang-radix-tree 补齐 c1/c2 池名: `unified_tree_core.rs` 的 `PoolName` 新增 `DeepseekV4C1/C1Indexer/C1IndexerScale` 与 `DeepseekV4C2/C2Indexer/C2IndexerScale`; `python_bindings.rs` 的 `pool_name_str()` 与 `parse_pool_name()` 同步新增 6 组字符串映射。两处必须成对修改。

4) sglang-server PD bootstrap 新增布局字段与一致性校验: `PrefillServerInfo` (响应, `skip_serializing_if`)、`Topology`、`Route` (请求, `serde(default)`) 三处加字段; `route_put` 在 `rcu` 闭包内判断 `registered_count > 0 && topo.dsv41_spec_layout != body.dsv41_spec_layout`, mismatch 时原样返回 `topo` (不递增计数、不写 `prefill_ranks`), 闭包外返回 400; `route_get` 的 sentinel 分支回传该字段。

5) 测试配套: 无独立测试文件, 但在 `bootstrap.rs` 的 `mod tests` 内联新增 `dspark_layout_round_trip_and_rank_mismatch`, 覆盖 PUT 200 → GET round trip → 不兼容 PUT 返回 400 且已存布局不变三条路径。

rust/sglang-radix-tree/src/python_bindings.rs

`PoolName` 与 Python 字符串的双向映射表, 新增 6 个 c1/c2 池名条目; 与 `unified_tree_core.rs` 的枚举变体必须成对修改。

```
/// 把 Rust 的 PoolName 枚举转成 Python 侧使用的字符串标识。
fn pool_name_str(name: PoolName) -> &'static str {
    match name {
        PoolName::Kv => "kv",
        PoolName::Mamba => "mamba",
        PoolName::Swa => "swa",
        PoolName::Indexer => "indexer",
        PoolName::DeepseekV4C4 => "deepseek_v4_c4",
        PoolName::DeepseekV4C4Indexer => "deepseek_v4_c4_indexer",
        PoolName::DeepseekV4C4IndexerScale => "deepseek_v4_c4_indexer_scale",
        PoolName::DeepseekV4C128 => "deepseek_v4_c128",
        // 新增: V4.1 的 c1 / c2 KV 池及其中间态池。
        PoolName::DeepseekV4C1 => "deepseek_v4_c1",
        PoolName::DeepseekV4C1Indexer => "deepseek_v4_c1_indexer",
```

```

PoolName::DeepseekV4C1IndexerScale => "deepseek_v4_c1_indexer_scale",
PoolName::DeepseekV4C2 => "deepseek_v4_c2",
PoolName::DeepseekV4C2Indexer => "deepseek_v4_c2_indexer",
PoolName::DeepseekV4C2IndexerScale => "deepseek_v4_c2_indexer_scale",
PoolName::DeepseekV4C4State => "deepseek_v4_c4_state",
PoolName::DeepseekV4C4IndexerState => "deepseek_v4_c4_indexer_state",
PoolName::DeepseekV4C128State => "deepseek_v4_c128_state",
PoolName::Draft => "draft",
PoolName::DraftIndexer => "draft_indexer",
PoolName::DraftSwa => "draft_swa",
}
}

/// 把 Python 传来的字符串反解为 Rust 枚举；未知名字报 PyValueError。
/// 反向映射与 pool_name_str 必须保持严格对称，否则会出现“能输出但读不回”的契约裂缝。
fn parse_pool_name(name: &str) -> PyResult<PoolName> {
    match name {
        "kv" => Ok(PoolName::Kv),
        "mamba" => Ok(PoolName::Mamba),
        "swa" => Ok(PoolName::Swa),
        "indexer" => Ok(PoolName::Indexer),
        "deepseek_v4_c4" => Ok(PoolName::DeepseekV4C4),
        "deepseek_v4_c4_indexer" => Ok(PoolName::DeepseekV4C4Indexer),
        "deepseek_v4_c4_indexer_scale" => Ok(PoolName::DeepseekV4C4IndexerScale),
        "deepseek_v4_c128" => Ok(PoolName::DeepseekV4C128),
        // 新增：与 pool_name_str 一一对应的 c1 / c2 系列。
        "deepseek_v4_c1" => Ok(PoolName::DeepseekV4C1),
        "deepseek_v4_c1_indexer" => Ok(PoolName::DeepseekV4C1Indexer),
        "deepseek_v4_c1_indexer_scale" => Ok(PoolName::DeepseekV4C1IndexerScale),
        "deepseek_v4_c2" => Ok(PoolName::DeepseekV4C2),
        "deepseek_v4_c2_indexer" => Ok(PoolName::DeepseekV4C2Indexer),
        "deepseek_v4_c2_indexer_scale" => Ok(PoolName::DeepseekV4C2IndexerScale),
        "deepseek_v4_c4_state" => Ok(PoolName::DeepseekV4C4State),
        "deepseek_v4_c4_indexer_state" => Ok(PoolName::DeepseekV4C4IndexerState),
        "deepseek_v4_c128_state" => Ok(PoolName::DeepseekV4C128State),
        "draft" => Ok(PoolName::Draft),
        "draft_indexer" => Ok(PoolName::DraftIndexer),
        "draft_swa" => Ok(PoolName::DraftSwa),
        other => Err(PyValueError::new_err(format!("unknown pool name: {other}"))),
    }
}

```

rust/sglang-radix-tree/src/unified_tree_core.rs

PoolName 枚举本身新增 6 个 c1/c2 变体，是 KV 池标识的数据契约源头，下游匹配与映射都依赖它。

```

/// 作为 PoolTransfer 标识的知名池名。
/// 注意：这是跨语言契约的源头——Rust 侧新增变体后，
/// python_bindings.rs 的字符串映射以及 Python 侧的池注册逻辑需同步跟进。

```

```
#[derive(Copy, Clone, PartialEq, Eq, Hash, Debug, Default)]
pub enum PoolName {
    #[default]
    Kv,
    Mamba,
    Swa,
    Indexer,
    DeepseekV4C4,
    DeepseekV4C4Indexer,
    DeepseekV4C4IndexerScale,
    DeepseekV4C128,
    // 新增: V4.1 的 c1 / c2 系列池 (含 indexer 与 indexer_scale 三种粒度) 。
    DeepseekV4C1,
    DeepseekV4C1Indexer,
    DeepseekV4C1IndexerScale,
    DeepseekV4C2,
    DeepseekV4C2Indexer,
    DeepseekV4C2IndexerScale,
    DeepseekV4C4State,
    DeepseekV4C4IndexerState,
    DeepseekV4C128State,
    Draft,
    DraftIndexer,
    DraftSwa,
}
```

rust/sglang-mm/src/lib.rs

dsv41 模块的接线入口：声明模块并在 PyO3 pymodule 中注册，使其在 python feature 下可被 Python 侧调用。

```
//! slang-mm: 面向 SGLang 的 Rust 加速多模态预处理。
//!
//! 以两种方式构建:
//! * PyO3 扩展 `sglang.srt.rust_extensions._multimodal` (feature `python`) ,
//! 供 Python 处理器 (如 Inkling) 与 parity 测试使用。
//! * 纯 Rust `rlib` (`default-features = false`) , 由 `sglang-server` 的
//! MM worker 路径链接——该依赖图内不含 pyo3。

pub mod common;
pub mod driver;
// 新增: V4.1 图像预处理模块 (resize + pad + patchify) 。
pub mod dsv41;
pub mod inkling;
pub mod pipeline;
pub mod qwen_vl;
pub mod registry;

#[cfg(feature = "python")]
use pyo3::prelude::*;
```

```
// 仅在 python feature 下导出；纯 Rust rlib 构建时不引入 pyo3。
#[cfg(feature = "python")]
#[pymodule]
fn _multimodal(m: &Bound<'_, PyModule>) -> PyResult<> {
    common::register(m)?;
    inkling::register(m)?;
    // 注册为新子模块 "dsv41", 与既有处理器并列。
    dsv41::register(m)?;
    qwen_vl::register(m)?;
    Ok(())
}
```

评论区精华

本 PR 的 review 区为空 (review_comments_count = 0, comments 仅一条来自作者的 issue 评论) , 因此没有技术性交锋。可观测到的两条信息：

作者在 PR body 中: "Landing this separately keeps the prebuilt-Rust-extension cache key stable for the Python-side changes that follow: the key hashes rust/**, so any PR carrying these files misses the cache and recompiles the extensions on every push."

作者在 issue 评论中: /tag-and-rerun-ci

后者的背景是 PR body 中 CI 状态显示 Latest PR Test (Base) 与 Latest PR Test (Extra) 均失败、AMD 无运行记录。材料未提供失败日志，无法判断失败是否与本 PR 改动相关。

风险与影响

风险点	位置	说明
首个注册 rank 定义布局契约	bootstr ap.rs route_ put	判定条件是 registered_count > 0, 第 1 个注册者无条件设定布局。若首个请求恰好不带 dsv41_spec_layout (None) , 后续真正带布局的 rank 会被判 mismatch 并 400, 可能导致整个 prefill 集群注册失败。代码未区分 " 未声明 " 与 " 声明为空 "。
rcu 闭包内的可变捕获	bootstr ap.rs route_ put	layout_mismatch 由闭包赋值。既有注释已说明 rcu 可能重跑闭包, 该标志依赖 " 最后一次执行结果 ", 并发注册下标志与最终提交状态理论上可能不同轮次。
跨语言枚举隐式契约	unified_ tree_ core.rs + pyth on_bin dings.r s	PoolName 是数据契约源头, 字符串映射分散两文件。本 PR 已同步 , 但未见穷举 round trip 单测; 后续新增变体仍有漏改风险, 漏改表现为运行期 unknown pool name。

风险点	位置	说明
新代码缺 parity 测试	dsv41/mod.rs	resize_patchify 校验较为完备 (checked_mul、几何约束、连续性检查)，但 PR 未附带与 Python/PIL 参考实现的数值 parity 测试，真正的等价性要靠后续 Python 侧接入 PR 验证。
CI 首轮失败	PR body	Base/Extra 两个 PR Test 失败，材料无日志，需以重跑结果为准。

影响面：改动落于多模态预处理、KV 池命名契约、PD 拆分引导三条链路，均为 V4.1 前置依赖。旧模型路径不受影响——新增模块、枚举变体与可选字段在旧路径上均不被触发。对用户而言，收益要等 Python 侧改动落地后才可见：图像预处理走 Rust 加速路径、HiCache 能识别 c1/c2 池、PD 启动期即可拒绝布局不一致的配置。

关联脉络

本 PR 是 DeepSeek-V4.1 支持序列中的 Rust 侧第一棒，与近期多条 PR 线相关：

V4/V4.1 模型专项：PR#38176 (DeepSeek-V4 确定性推理下 RouteGEMM 保持 p32) 与本 PR 同属 DeepSeek 系列的正确性保障投入。

- PD 拆分：PR#38984 (INTRA_NODE_NVLINK 全层合并单批，KV 传输延迟降 33%) 关注传输性能，本 PR 关注 PD 启动期的布局一致性校验，两者是 V4 系列在 disaggregation 场景的不同侧面。
- 统一基数树与 HiCache：PR#39283 修改了 rust/sglang-radix-tree/src/unified_tree_core.rs 与 Python 侧 adapter，正是本 PR 扩展 PoolName 所在的文件；PR#39480 优化了 HiCache buffer 模式的存在性缓存。三者叠加说明统一基数树的池契约正处于持续演进中。
- Rust 前端扩展：PR#39329 (为 Rust 前端新增外部多模态处理器扩展点) 与本 PR 在 sglang-server / sglang-mm 两侧同时扩展能力的模式一致，反映仓库正在把多模态与服务器侧逻辑逐步下沉到 Rust。

按 PR body 的说明，后续会有一个 "Python 侧改动 " 的 PR 紧随其后，把 resize_patchify、deepseek_v4_c1/c2 池名与 dsv41_spec_layout 真正接入到 Python 的处理链路与客户端中，届时才是这套 Rust 接口的真正验证点。