PR #51178 完整报告

vllm-project/vllm

[Rust Frontend][gRPC] Add explicit data-parallel rank routing

合并时间：2026-08-11 06:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51178

执行摘要

本 PR 为 Rust frontend 的 gRPC 推理 API 新增显式 data-parallel rank 路由：客户端通过 x-data-parallel-rank metadata 即可把请求精确投递到指定全局 DP 副本，未设置时保持原有负载均衡。实现上把引擎身份编码收紧为 u16、从类型层面杜绝静默截断，并在前端新增 data_parallel_size 配置与校验，使拓扑上报（server_info、world_size）与引擎连接范围一致。这是 Rust 前端路由语义从"本地索引"升级为"全局身份"的关键一步，讨论中关于 metadata 设计、DP size 归属与能力字段取舍的决策对后续协议演进有直接借鉴价值。

功能与动机

PR body 的 Purpose 明确说明动机："Add explicit data-parallel rank routing to the Rust frontend gRPC inference API"。具体诉求有三点：

- 请求可显式携带全局 DP rank，并在 request conversion 后用于 EngineCore 选择；
- 未设置 rank 时保留负载均衡，已设置时只路由到全局身份匹配的已连接引擎；
- "Reject ranks that cannot be represented by vLLM's two-byte engine identity instead of allowing integer truncation to alias another rank"——拒绝而非截断，防止 rank 别名路由到错误引擎。

在分体推理、KV 亲和路由等场景下，调用方需要确定性路由能力，而 Rust 前端此前只能做本地索引 + 负载均衡。njhill 在评论中还坚持 rank 应走 gRPC metadata 而非 proto 字段（引用早期 PR #48033 的讨论），这也是最终方案的重要约束。

实现拆解

1. gRPC 入口接收 rank（rust/src/server/src/grpc/inference.rs） - 新增 DATA_PARALLEL_RANK_METADATA_KEY = "x-data-parallel-rank" 常量与 data_parallel_rank_from_metadata() 解析函数。 - generate 与 generate_stream 两个 RPC 在 prepare_request 前解析 metadata，并写入 TextRequest.data_parallel_rank。 - 之所以用 metadata 而非 proto 字段，是采纳 njhill 在 #48033 评论中的建议，避免路由信息入侵请求 schema。
2. 核心路由按全局身份匹配（rust/src/engine-core-client/src/client/state.rs + transport.rs） - choose_engine_for_request 在指定 rank 时先 u16::try_from(rank) 防截断，再通过 routing_per_engine.contains_key() 校验引擎已连接，错误信息携带 connected_ranks 全量列表。 - EngineId::from_engine_index 签名从 u32 收紧为 u16；

apply_scheduler_counts 增加 u16 防护；connect_bootstrapped 对 engine_start_index + engine_count 做区间边界校验。

3. 部署级 DP 拓扑持有与校验（config.rs + state.rs + control.rs、world_size.rs） - Config 新增 data_parallel_size 字段，validate() 校验其 ≥ 1、≤ u16::MAX + 1，并分别对 HandshakeOwner（engine_count 必须相等）、Bootstrapped（引擎范围不得越界）做交叉验证。 - AppState 新增对应字段与 with_data_parallel_size() 注入方法；server_info 与 world_size 上报改用配置值，并删除 EngineCoreClient::data_parallel_size() 死代码。 - Python 侧 serve.py + vllm/v1/utils.py 新增 --data-parallel-size 参数透传。
4. 配套收尾 - mock-engine 的 run_engine 适配 u16 限制（usize 参数 + 内部 try_from）。 - InvalidDataParallelRank 错误重构为 connected_ranks: Vec<u32>。 - 测试以内嵌 Rust 测试为主：新增 unary_generate_unconnected_data_parallel_rank_returns_invalid_argument（全局 rank 3 引擎拒绝 rank 0）、register_with_rank_uses_global_engine_identity（全局身份语义）、control_aggregates_multi_engine_capacity 扩展（server_info 上报 DP size）。PR body 确认 GPU 验证覆盖聚合、DP2 KV 亲和路由、分体推理、多模态、KVBM、取消隔离与引擎故障恢复。

关键源码片段

rust/src/engine-core-client/src/client/state.rs

核心路由逻辑所在：choose_engine_for_request 实现全局 DP rank 精确匹配与 u16::try_from 防截断，错误通过 connected_ranks 列表报告；apply_scheduler_counts 增加 u16 防护。

```
/// 为请求选择目标引擎：未指定 rank 时按负载均衡，指定时按全局
/// data-parallel rank 精确匹配已连接引擎的身份。
fn choose_engine_for_request(&mut self, data_parallel_rank: Option<u32>) -> Result<EngineId>
{
    if let Some(rank) = data_parallel_rank {
        // vLLM EngineCore 的 ROUTER/DEALER 身份是两字节小端编码，
        // 先把 u32 rank 无损收敛到 u16；否则直接视为无效 rank，
        // 防止整数截断后路由到身份编码恰好相同的另一个引擎。
        let engine_id = u16::try_from(rank).ok().map(EngineId::from_engine_index);
        // 同时要求“编码成功”且“该引擎确实连接到了本前端”，
        // 二者缺一都返回 InvalidDataParallelRank。
        return engine_id
            .filter(|engine_id| self.routing_per_engine.contains_key(engine_id))
            .ok_or_else(|| Error::InvalidDataParallelRank {
                rank,
                // 错误信息带上已连接 rank 的全集，调用方可以区分
                // “rank 写错”和“目标引擎还没连上”两种情况。
                connected_ranks: self
                    .routing_per_engine
                    .keys()
                    .filter_map(EngineId::engine_index)
                    .collect(),
            });
```

```
    }

    // 未指定 rank：沿用负载均衡，选择路由得分最小的引擎。
    Ok(self
        .routing_per_engine
        .iter()
        .min_by_key(|(_, state)| state.routing_score())
        .map(|(engine_id, _)| engine_id.clone())
        .expect("request registry must contain at least one engine"))
}
```

rust/src/server/src/grpc/inference.rs

gRPC 入口：新增 x-data-parallel-rank metadata 常量与 data_parallel_rank_from_metadata()，在 generate / generate_stream 两个 RPC 中解析并注入 TextRequest.data_parallel_rank。

```
/// gRPC metadata 键名：请求显式指定 data-parallel rank。
///
/// rank 属于“如何路由这次调用”的传输层信息，刻意不放进 `GenerateRequest`
/// proto 结构体，避免路由语义入侵请求 schema（njhill 在早期 PR #48033
/// 中坚持的设计）。
const DATA_PARALLEL_RANK_METADATA_KEY: &str = "x-data-parallel-rank";

/// 解析 metadata 中的可选 rank：缺省返回 `None`（走负载均衡），
/// 值必须是可解析为 u32 的字符串，否则以 `InvalidArgument` 拒绝，
/// 避免非法输入被静默忽略。
fn data_parallel_rank_from_metadata(
    request: &Request<pb::GenerateRequest>,
) -> Result<Option<u32>, Status> {
    let Some(value) = request.metadata().get(DATA_PARALLEL_RANK_METADATA_KEY) else {
        return Ok(None);
    };
    let value = value.to_str().map_err(|_| {
        Status::invalid_argument("x-data-parallel-rank metadata must be an unsigned 32-bit
        integer")
    })?;
    value.trim().parse::<u32>().map(Some).map_err(|_| {
        Status::invalid_argument("x-data-parallel-rank metadata must be an unsigned 32-bit
        integer")
    })
}
```

rust/src/engine-core-client/src/transport.rs

EngineId::from_engine_index 从 u32 收紧为 u16，类型层面杜绝截断；connect_bootstrapped 对 engine_start_index + engine_count 增加两字节身份区间校验。

```
if engine_count == 0 {
    bail_unexpected_handshake_message!("expected engine_count >= 1");
}
```

```
// bootstrapped 模式的引擎身份由 [engine_start_index, +engine_count)
// 连续合成，必须先验证整个区间落在两字节引擎身份范围内，否则
// EngineId 编码会静默回绕，导致路由别名。
let engine_start_index =
    u16::try_from(engine_start_index).map_err(|_| Error::UnexpectedHandshakeMessage {
        message: "engine_start_index exceeds the two-byte engine identity limit".to_string(),
    })?;
let engine_end_index =
    usize::from(engine_start_index).checked_add(engine_count).ok_or_else(|| {
        Error::UnexpectedHandshakeMessage {
            message: "engine_start_index + engine_count overflows".to_string(),
        }
    })?;
if engine_end_index > usize::from(u16::MAX) + 1 {
    return Err(Error::UnexpectedHandshakeMessage {
        message: "engine_start_index + engine_count exceeds the two-byte engine identity limit"
            .to_string(),
    });
}
```

评论区精华

> “I still feel that this would be better as a grpc metadata field as explained in my comment on the earlier PR #48033”——njhill 在 issue 评论中坚持 metadata 方案，作者最终据此重构。

> “Theoretically the data parallel size should have been passed from the engine to the frontend during handshake via EngineCoreReadyResponse ... opening a PR for this: #51245”——BugenZhao 指出握手传递的 DP size 对 dense 模型不准确。

> “I'm actually not sure about this. I think it might actually better for now to do something along the lines of what @connorcarpenter15 had originally ... follow what the python front-end does where it retains the externally configured DP size but the engines themselves aren't aware of this and behave as DP=1.”——njhill 支持保留前端配置值，作者确认“前端是唯一访问配置 DP size 的地方，引擎处处按 DP=1 处理”。

> “Personally, I don't think we need to maintain backward compatibility or worry about rolling upgrades at this stage, since it's still very early. Or can we detect this by checking the engine_version”——BugenZhao 反对能力 flag，最终 supports_explicit_data_parallel_rank 被移除。

> “This is a good catch! Looks like something missed when adding hybrid/external DPLB.”——BugenZhao 对 connected_ranks 错误信息的评价。

风险与影响

- metadata 解析 fail-loud：data_parallel_rank_from_metadata 对非 u32 字符串直接返回 InvalidArgument，调用方一旦携带非法值会立即触发新错误路径。

- 类型收紧联动面：EngineId::from_engine_index 改 u16 波及 mock-engine、测试与调度统计路径；apply_scheduler_counts 对超范围索引静默忽略（返回 false），存在监控盲区。
- 配置一致性假设：config.rs 对 HandshakeOwner 模式强制 engine_count == data_parallel_size，对多前端共享引擎、非连续 rank 的部署形态会直接拒绝启动。
- 协议无前后兼容承诺：BugenZhao 明确建议早期阶段不维护滚动升级兼容，客户端与 server 需同步升级。
- 影响范围：调用方获得确定性路由能力；前端路由语义与 Python 前端 DP 语义对齐；/world_size 上报语义从引擎握手值改为前端配置值，多前端聚合场景需注意。

关联脉络

本 PR 处于 Rust frontend 数据并行能力演进的中间节点：

- #48033（njhill 评论引用）：早期 PR 确立了“DP rank 走 gRPC metadata”的设计方向；
- #51245（BugenZhao 在讨论中开出）：计划通过握手 EngineCoreReadyResponse 传递真实 DP size，修复 dense 模型覆盖问题，本 PR 的 data_parallel_size 配置字段预计在该 PR 落地后迁移，二者是承接关系；
- 同仓库 #51276（Buf 发布 gRPC protobuf schema）属于同一 Rust gRPC 基础设施线，本 PR 的 metadata 设计不依赖 proto 变更，恰恰降低了 schema 发布后的演进成本。

整体看，Rust 前端正在逐步对齐 Python 前端的分布式语义，DP rank 路由是其中关键一步。