PR #25083 完整报告

sgl-project/sglang

fix(mooncake): honour MOONCAKE_PROTOCOL so EFA hardware can select efa transport

合并时间：2026-05-29 14:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25083

执行摘要

- 一句话：Mooncake 传输引擎支持环境变量选择 EFA 协议
- 推荐动作：建议精读本 PR 的代码改动，重点关注 " 如何通过单一环境变量统一两个调用点的协议配置 " 以及 " 重构后分支合并的代码组织 "。对于部署在 AWS EFA 的用户，该 PR 是必合的。

功能与动机

在 AWS EFA 实例（如 p5en.48xlarge）上运行 SGLang 时，使用 --disaggregation-transfer-backend mooncake 会在首次 KV 缓存传输时崩溃，错误为 " Failed to create QP: Operation not supported [95]"。根本原因是两个调用点硬编码了 " rdma" 协议，而 EFA 硬件需要 "efa" 协议（使用 libfabric SRD，而非 ibverbs RC QP）。详情见关联 Issue #24838。

实现拆解

1. 修改默认协议：在 python/sglang/srt/environ.py 中，将 MOONCAKE_PROTOCOL 的默认值从 "tcp" 改为 "rdma"，使所有消费者（包括 Mooncake Store 和传输引擎）默认使用 rdma 协议，保持向后兼容。
2. 重构传输引擎初始化：在 python/sglang/srt/distributed/device_communicators/mooncake_transfer_engine.py 的 initialize 方法中，移除了硬编码的 "rdma" 分支，改用 envs.MOONCAKE_PROTOCOL.get() 读取协议字符串；同时将 Ascend 和非 Ascend 路径合并为一条 engine.initialize 调用，提高了代码可维护性。
3. 适配模型运行器：在 python/sglang/srt/model_executor/model_runner.py 的 remote_instance_init_transfer_engine 方法中，同样将硬编码的 "rdma" 替换为 envs.MOONCAKE_PROTOCOL.get()，确保远程实例传输引擎也能响应环境变量。

该变更不涉及测试文件，因为端到端 EFA 测试需要特殊硬件和编译环境。环境变量 MOONCAKE_PROTOCOL 已在 Mooncake Store 的 README 中。

关键文件：

- python/sglang/srt/distributed/device_communicators/mooncake_transfer_engine.py（模块 传输引擎；类别 source；类型 core-logic；符号 initialize）：核心修改：移除硬编码 rdma，改用 envs.MOONCAKE_PROTOCOL 读取协议，并重构 Ascend 分支逻辑，使通过单一初始化路径运行。

- python/sglang/srt/model_executor/model_runner.py（模块 模型运行；类别 source；类型 core-logic；符号 remote_instance_init_transfer_engine）：第二个调用点，同样使用 envs.MOONCAKE_PROTOCOL 替换硬编码 rdma，确保远程实例传输引擎初始化一致。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration；符号 MOONCAKE_PROTOCOL）：修改 MOONCAKE_PROTOCOL 默认值从 tcp 为 rdma，与所有消费者形成一致，避免 Mooncake Store 用户使用错误协议。

关键符号：MooncakeTransferEngine.initialize, ModelRunner.remote_instance_init_transfer_engine, MOONCAKE_PROTOCOL

关键源码片段

python/sglang/srt/distributed/device_communicators/mooncake_transfer_engine.py

核心修改：移除硬编码 rdma，改用 envs.MOONCAKE_PROTOCOL 读取协议，并重构 Ascend 分支逻辑，使通过单一初始化路径运行。

```
def initialize(
    self,
    hostname: str,
    device_name: Optional[str],
) -> None:
    """Initialize the mooncake instance."""
    # 当启用了 Ascend 传输时，使用 "ascend" 协议
    if envs.ENABLE_ASCEND_TRANSFER_WITH_MOONCAKE.get():
        npu_phy_id = envs.ASCEND_NPU_PHY_ID.get()
        suffix = self.gpu_id if npu_phy_id == -1 else npu_phy_id
        hostname += f":{get_free_port()}:npu_{suffix}"
        protocol = "ascend"
    else:
        # MOONCAKE_PROTOCOL 选择传输层协议：rdma | efa | tcp | ...
        # 默认是 "rdma"；在 AWS EFA 上要设置 MOONCAKE_PROTOCOL=efa
        protocol = envs.MOONCAKE_PROTOCOL.get()

    # 将 protocol 传递给 Mooncake TransferEngine
    ret_value = self.engine.initialize(
        hostname,
        "P2PHANDSHAKE",
        protocol,
        device_name if device_name is not None else "",
    )
    if ret_value != 0:
        logger.error("Mooncake Transfer Engine initialization failed.")
        raise RuntimeError("Mooncake Transfer Engine initialization failed.")
```

python/sglang/srt/model_executor/model_runner.py

第二个调用点，同样使用 envs.MOONCAKE_PROTOCOL 替换硬编码 rdma，确保远程实例传输引擎初始化一致。

```
def remote_instance_init_transfer_engine(self):
    try:
        from mooncake.engine import TransferEngine
    except ImportError as e:
        logger.warning(
            "Please install mooncake for using remote instance transfer engine: pip install
            mooncake"
        )
        return
    self.remote_instance_transfer_engine = TransferEngine()
    local_ip = get_local_ip_auto()
    # 使用 envs.MOONCAKE_PROTOCOL.get() 获取协议（默认 "rdma"），
    # 用户可通过环境变量覆盖（例如 export MOONCAKE_PROTOCOL=efa）
    self.remote_instance_transfer_engine.initialize(
        local_ip,
        "P2PHANDSHAKE",
        envs.MOONCAKE_PROTOCOL.get(),
        envs.MOONCAKE_DEVICE.get(),
    )
    self.remote_instance_transfer_engine_session_id = NetworkAddress(
        local_ip, self.remote_instance_transfer_engine.get_rpc_port()
    ).to_host_port_str()
```

评论区精华

1. 使用环境变量管理方案：gemini-code-assist[bot] 建议使用集中的 envs 对象而非直接访问 os.environ。作者回复称已采纳，并改用 envs.MOONCAKE_PROTOCOL.get()，同时通过 is_set() 守卫保留旧行为。
2. 默认值争议：ShangmingCai 指出需要将默认值改为 "rdma" 而非 "tcp"，以避免现有 IB/RoCE 用户意外降级。作者最终在后续 commit 中将 environ.py 中的默认值改为 "rdma"，并移除了 is_set() 守卫。
3. 逻辑分支合并建议：stmatengss 建议合并 Ascend 和通用分支的初始化逻辑，使代码更简洁。作者重构后使用单一 protocol 变量，统一调用 engine.initialize。
4. Lint 修复与 CI 重跑：ShangmingCai 要求修复 lint，并多次使用 /rerun-failed-ci 命令重跑失败的 CI（最终多为 infra 问题）。

- 环境变量访问方式 (design): 作者接受建议，改用 envs.MOONCAKE_PROTOCOL.get()，并添加 is_set() 守卫保留旧行为。
- 合并逻辑分支 (style): 作者重构 initialize 方法，使用单一 protocol 变量，所有分支共用同一个 engine.initialize 调用。
- 默认值变更 (correctness): 作者将 environ.py 中的默认值改为 "rdma"，并移除了 is_set() 守卫。

风险与影响

- 风险：
  1. 回归风险：现有 InfiniBand/RoCE 用户若不设置 MOONCAKE_PROTOCOL，默认仍为 "rdma"，与之前行为一致；Mooncake Store 用户之前依赖默认 "tcp"，但 Store 有自己的初始化逻辑，且团队已确认该默认值变更对 Store 无影响。
1. 兼容性风险：用户需使用编译了 USE_EFA=ON 的 Mooncake wheel 才能使用 "efa" 协议，否则初始化仍会失败。
2. 缺少测试覆盖：没有对应的端到端测试，因为需要 EFA 硬件和定制编译。CI 中仅依赖基础单元测试，可能遗漏回归。 - 影响：用户影响：AWS EFA 用户现在可以通过设置 MOONCAKE_PROTOCOL=efa 来启用 EFA 传输，消除 KV 缓存传输崩溃。现有 IB/RoCE 用户无感知。

系统影响：变更限于初始化阶段，无运行时性能影响。

团队影响：低维护成本，环境变量方案统一了协议配置方式。

- 风险标记：默认值变更影响 Mooncake Store 使用者，需要 Mooncake 编译支持 EFA，缺少端到端测试覆盖

关联脉络

- PR #24846 [Bug] Mooncake transfer backend crashes on AWS EFA: hardcoded rdma protocol bypasses efa_transport: 同一 bug 的早期尝试，作者关闭后重新发起本 PR。