PR #41980 完整报告

vllm-project/vllm

use split_group for pytorch process group creation

合并时间：2026-06-05 02:36

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41980

执行摘要

- 一句话：使用 split_group 替代 new_group 创建分布式子组
- 推荐动作：值得精读，特别是分布式初始化细节和 split_group 的性能优势。设计上采用环境变量作为 rollback 机制、分离新旧路径的做法值得借鉴。合并后应关注后续启用 PR 的测试结果。

功能与动机

PyTorch 正在废弃 lazy-init 路径，推荐 *eager* init；split_group 比 new_group 更高效（复用父通信器 bootstrap）；torchcomms 只支持 split_group，采用 split_group 使得后续切换到 torchcomms 只需修改环境变量。

实现拆解

1. 在 vllm/envs.py 中新增环境变量 VLLM_DISTRIBUTED_USE_SPLIT_GROUP（默认 False），控制是否使用新路径。
2. 在 vllm/distributed/parallel_state.py 中新增辅助函数：_platform_device_type（返回当前平台设备类型）、_device_backend_str（将后端名标准化为 <device>:<backend> 格式）、_create_subgroups_split_group（通过 split_group 创建设备和 CPU 子组）、_init_process_group_for_split_group（初始化默认 PG 时使用混合后端和 device_id 绑定）、_validate_default_pg_for_split_group（验证默认 PG 是否满足 split_group 要求）。
3. 修改 GroupCoordinator.__init__：根据 VLLM_DISTRIBUTED_USE_SPLIT_GROUP 选择调用 _create_subgroups_split_group 或原有 new_group 逻辑。
4. 修改 init_distributed_environment：当启用 split_group 时，使用 cpu:gloo,cuda:nccl 混合后端并传递 device_id，确保 *eager* init。
5. 新增 tests/distributed/test_split_group.py 专门测试 split_group 路径，并在 test_pynccl.py、test_quick_all_reduce.py、test_torchrun_example.py 等测试中添加条件分支，以兼容两种路径。

关键文件：

- vllm/distributed/parallel_state.py（模块 分组管理；类别 source；类型 dependency-wiring；符号 _platform_device_type, _device_backend_str, _create_subgroups_split_group, _init_process_group_for_split_group）：核心变更文件，新增 split_group 相关的辅助函数并修改 GroupCoordinator 初始化逻辑。

- tests/distributed/test_split_group.py（模块 测试；类别 test；类型 test-coverage；符号 distributed_run, worker_fn_wrapper, wrapped_fn, _verify_device_group）：新增测试文件，专门覆盖 split_group 路径的基本功能和多子组场景。
- vllm/envs.py（模块 环境变量；类别 source；类型 core-logic）：新增环境变量 VLLM_DISTRIBUTED_USE_SPLIT_GROUP 作为特性开关。
- tests/distributed/test_pynccl.py（模块 测试；类别 test；类型 test-coverage）：修改测试以支持 split_group 路径，添加条件分支。
- tests/distributed/test_quick_all_reduce.py（模块 测试；类别 test；类型 test-coverage）：修改测试以支持 split_group 路径，调整 init_process_group 调用。
- tests/distributed/test_torchrun_example.py（模块 测试；类别 test；类型 test-coverage）：修改 torchrun 示例测试，展示 split_group 所需的 init_process_group 新写法。
- tests/distributed/test_torchrun_example_moe.py（模块 测试；类别 test；类型 test-coverage）：与 test_torchrun_example.py 类似的修改。

关键符号：_platform_device_type, _device_backend_str, _create_subgroups_split_group, _init_process_group_for_split_group, _validate_default_pg_for_split_group, GroupCoordinator.init, init_distributed_environment

关键源码片段

vllm/distributed/parallel_state.py

核心变更文件，新增 split_group 相关的辅助函数并修改 GroupCoordinator 初始化逻辑。

```
# vllm/distributed/parallel_state.py

def _platform_device_type() -> str:
    """Return the device-type string (e.g. "cuda", "xpu", "cpu")
    for the current platform, in the form expected by
    ``torch.distributed.init_process_group(backend=...)``.
    """
    from vllm.platforms import current_platform

    if current_platform.is_cuda_alike():
        return "cuda"
    elif current_platform.is_xpu():
        return "xpu"
    elif current_platform.is_out_of_tree():
        return current_platform.device_name
    else:
        return "cpu"


def _device_backend_str(torch_distributed_backend: str | Backend) -> str:
    """Normalize ``torch_distributed_backend`` to the ``"<device>:<backend>"``
    format required by ``split_group``'s ``backend`` argument.
    """
```

```
    backend_str = str(torch_distributed_backend)
    if ":" in backend_str:
        return backend_str
    return f"{_platform_device_type()}:{backend_str}"


def _create_subgroups_split_group(
    group_ranks: list[list[int]],
    group_name: str,
    torch_distributed_backend: str | Backend,
) -> tuple[ProcessGroup, ProcessGroup]:
    """Create the device + CPU subgroups for ``GroupCoordinator`` via
    ``torch.distributed.split_group``.

    ``split_group`` is collective on the parent group, so every parent rank
    must enter with the same ``split_ranks`` definition. Each rank receives
    the subgroup it belongs to.
    """
    device_backend_str = _device_backend_str(torch_distributed_backend)
    self_device_group = torch.distributed.split_group(
        split_ranks=group_ranks, # pass full group_ranks to meet collective requirement
        group_desc=f"{group_name}:device",
        backend=device_backend_str,
    )
    # CPU subgroup: split_group requires the requested backend filter to
    # include the parent's default device type (the device the parent PG
    # was bound to via ``device_id``), so a cpu-only filter is rejected.
    # Include the device backend in the filter; only the gloo backend is
    # actually used for CPU collectives on this group.
    self_cpu_group = torch.distributed.split_group(
        split_ranks=group_ranks,
        group_desc=f"{group_name}:cpu",
        backend=f"cpu:gloo,{device_backend_str}",
    )
    return self_device_group, self_cpu_group
```

tests/distributed/test_split_group.py

新增测试文件，专门覆盖 split_group 路径的基本功能和多子组场景。

```
# tests/distributed/test_split_group.py

import os
import multiprocess as mp
import pytest
import torch
import torch.distributed

import vllm.envs as envs
from vllm.distributed.parallel_state import (
```

```
    GroupCoordinator,
    init_distributed_environment,
)
from vllm.utils.system_utils import update_environment_variables

# The whole module exercises the split_group code path, which is opt-in
# behind VLLM_DISTRIBUTED_USE_SPLIT_GROUP=1.
pytestmark = pytest.mark.skipif(
    not envs.VLLM_DISTRIBUTED_USE_SPLIT_GROUP,
    reason=("VLLM_DISTRIBUTED_USE_SPLIT_GROUP=1 not set; split_group path is opt-in."),
)

mp.set_start_method("spawn", force=True)

def distributed_run(fn, world_size):
    # ... (setup env vars and spawn processes)...

def worker_fn_wrapper(fn):
    def wrapped_fn(env):
        update_environment_variables(env)
        local_rank = os.environ["LOCAL_RANK"]
        device = torch.device(f"cuda:{local_rank}")
        torch.accelerator.set_device_index(device)
        init_distributed_environment()
        fn()
    return wrapped_fn

def _verify_device_group(coordinator: GroupCoordinator):
    """Verify device group works via all-reduce."""
    tensor = torch.ones(16, 16, dtype=torch.float32, device=torch.device(f"cuda:{torch.
    distributed.get_rank()}"))
    torch.distributed.all_reduce(tensor, group=coordinator.device_group)
    torch.accelerator.synchronize()
    assert torch.all(tensor == coordinator.world_size).cpu().item()

def _verify_cpu_group(coordinator: GroupCoordinator):
    """Verify CPU group works via all-reduce."""
    tensor = torch.ones(16, dtype=torch.float32)
    torch.distributed.all_reduce(tensor, group=coordinator.cpu_group)
    assert torch.all(tensor == coordinator.world_size).cpu().item()
```

vllm/envs.py

新增环境变量 VLLM_DISTRIBUTED_USE_SPLIT_GROUP 作为特性开关。

```
# vllm/envs.py

# 在类定义中新增（第 66 行附近）：
VLLM_DISTRIBUTED_USE_SPLIT_GROUP: bool = False

# 在解析函数中新增（第 880 行附近）：
```

```
"VLLM_DISTRIBUTED_USE_SPLIT_GROUP": lambda: bool(
    int(os.getenv("VLLM_DISTRIBUTED_USE_SPLIT_GROUP", "0"))
),
```

评论区精华

- split_group 契约正确性（zou3519）：每个 rank 必须用相同的 split_ranks 调用，否则可能 hang。PR 修改后使用完整的 group_ranks 满足要求。
- 环境变量命名（kapilsh）：建议 VLLM_DISTRIBUTED_USE_SPLIT_GROUP 更明确，已采用。
- 代码重构建议（kapilsh）：建议将新逻辑提取为独立函数，PR 已实现 _create_subgroups_split_group 等函数。
- PyTorch版本依赖（zou3519）：需要PyTorch nightly（2.13+），CI中是否支持需确认。
- 测试分支死代码（zou3519）：其他测试文件中的 split_group 分支在默认关闭时是死代码，但保留了为后续启用时使用。
- split_group 调用契约 (correctness): tushar00jain 将调用改为传入完整的 group_ranks，满足契约。
- 环境变量命名 (design): 采用该建议。
- 代码组织建议 (design): 已提取 _create_subgroups_split_group 等函数。
- PyTorch 版本依赖 (question): tushar00jain 确认需要 nightly，但 vLLM 有 nightly CI 流程可以覆盖；后续需跟进。

风险与影响

- 风险：
  1. 核心路径变更：GroupCoordinator 是分布式通信的核心，新路径可能引入 hang 或语义错误（如 split_group 调用不一致）。
  2. 版本依赖风险：需要使用支持 split_group 的 PyTorch 版本（nightly），若 vLLM 仍在用较旧版本，该特性无法生效或需版本守卫。
  3. 测试覆盖不均衡：新路径默认关闭，现有 CI 不运行新测试文件，可能遗漏回归。
  4. 多个测试文件的条件分支：维护两份路径增加复杂度，将来启用后需确保所有分支行为一致。 - 影响：对用户：默认无影响，因为环境变量默认关闭；启用后提升大规模分布式初始化速度。对系统：核心子组创建方式改变，但通过环境变量向下兼容。对团队：需关注 PyTorch 版本兼容性和后续启用的回归测试，PR 为 torchcomms 迁移奠定了基础。 - 风险标记：核心路径变更，版本依赖风险，测试覆盖不均衡，条件分支维护成本

关联脉络

- PR #42471 (unknown, from stack): PR body 的 stack 中标记该 PR 为 #41980 的后继，可能启用 split_group 或进一步清理。
- PR #42565 (unknown, from stack): PR body 的 stack 中标记该 PR 为 #41980 的后继，可能启用 split_group 或进一步清理。