PR #6316 完整报告

verl-project/verl

[ci, trainer] fix: fix code, scripts and st for mindspeedllm backend

合并时间：2026-05-25 14:26

原文链接：http://prhub.com.cn/verl-project/verl/pull/6316

执行摘要

- 一句话：重命名 MindSpeed 策略并修复后端兼容性
- 推荐动作：建议精读，重点关注 MindSpeedEngineConfig 的变更和 get_base_mcore_config_from_engine_config 的设计取舍。合并后可引导用户迁移配置，并尽快补充 FSDP 策略的处理。

功能与动机

MindSpeed 后端在 verl 中需要统一策略标识和配置结构，以支持新的 mindspeed_megatron 和 mindspeed_fsdp 后端。旧命名 mindspeed_llm 已不再适用，且 llm_kwargs/mm_kwargs 需要重构为更通用的 mcore_kwargs。

实现拆解

1. Engine 配置类重命名：在 verl/workers/config/engine.py 中，将 MindSpeedEngineConfig 的默认策略从 mindspeed_llm 改为 mindspeed_megatron，并将字段 llm_kwargs/mm_kwargs 替换为 mcore_kwargs/fsdp_kwargs。
2. Engine 实现类重命名：在 verl/workers/engine/mindspeed/transformer_impl.py 中，将 MindSpeedLLMEngineWithLMHead 重命名为 MindSpeedMegatronEngineWithLMHead，并修复 _build_megatron_module 中 forward_only 分支的逻辑，移除早期 return 使代码更简洁。
3. 配置转换函数适配：在 verl/workers/engine/mindspeed/utils.py 中，get_base_mcore_config_from_engine_config 不再按策略分支更新 kwargs，而是统一通过 engine_config.mcore_kwargs 注入；同时引入 get_hf_rope_theta 替代直接取 int(hf_config.rope_theta)，以支持更多模型。
4. 导入导出符号更新：更新 verl/workers/engine/__init__.py 和 verl/workers/engine/mindspeed/__init__.py 中的导入和 __all__ 列表，引用重命名后的类。
5. 测试脚本与 CI 适配：新增 tests/special_npu/nightly_ci_ascend/run_grpo_qwen3_8b_mindspeedllm_npu.sh 作为 nightly CI 测试脚本，更新其他现有测试脚本中的策略名称和配置字段；修改 .github/workflows/nightly_ascend.yml 添加新测试 Job。
6. 清理旧示例脚本：删除不再使用的 examples/grpo_trainer/run_qwen3_8b_mindspeed.sh，并用 examples/ascend_extras/grpo_trainer/run_qwen3_30b_a3b_mindspeed.sh 替代。

关键文件：

- verl/workers/engine/mindspeed/transformer_impl.py（模块 引擎层；类别 source；类型 core-logic；符号 MindSpeedLLMEngineWithLMHead, MindSpeedMegatronEngineWithLMHead, _build_megatron_module）：核心 engine 实现文件，重命名类并修复 forward_only 分支逻辑。
- verl/workers/config/engine.py（模块 配置层；类别 source；类型 core-logic；符号 MindSpeedEngineConfig）：配置 dataclass 变更，重命名策略和 kwargs 字段。
- verl/workers/engine/mindspeed/utils.py（模块 工具函数；类别 source；类型 dependency-wiring；符号 get_base_mcore_config_from_model_config, get_base_mcore_config_from_engine_config）：配置转换函数适配新策略和 kwargs。
- verl/workers/engine/__init__.py（模块 引擎层；类别 source；类型 dependency-wiring）：更新导入符号，确保模块级引用正确。
- verl/workers/engine/mindspeed/__init__.py（模块 引擎层；类别 source；类型 dependency-wiring）：包内导出更新，确保模块符号可见。
- tests/special_npu/nightly_ci_ascend/run_grpo_qwen3_8b_mindspeedllm_npu.sh（模块 测试脚本；类别 test；类型 test-coverage）：新增 nightly CI 测试脚本，验证 Qwen3-8B GRPO 在 mindspeedllm 后端上的正确性。
- .github/workflows/nightly_ascend.yml（模块 CI 配置；类别 infra；类型 infrastructure）：CI 配置新增测试 Job，确保持续集成覆盖。

关键符号：MindSpeedLLMEngineWithLMHead -> MindSpeedMegatronEngineWithLMHead, _build_megatron_module, get_base_mcore_config_from_engine_config, get_base_mcore_config_from_model_config, MindSpeedEngineConfig.post_init

关键源码片段

verl/workers/engine/mindspeed/transformer_impl.py

核心 engine 实现文件，重命名类并修复 forward_only 分支逻辑。

```
# verl/workers/engine/mindspeed/transformer_impl.py

@EngineRegistry.register(model_type="language_model", backend="mindspeed_megatron",
device="npu")
class MindSpeedMegatronEngineWithLMHead(MegatronEngineWithLMHead):
    """MindSpeed Megatron 引擎实现，适用于语言模型（非值模型）。"""

    def __init__(
        self,
        model_config: HFModelConfig,
        engine_config: MindSpeedEngineConfig,
        optimizer_config: McoreOptimizerConfig,
        checkpoint_config: CheckpointConfig,
    ):
        super().__init__(model_config, engine_config, optimizer_config, checkpoint_config)

    def _init_device_mesh(self):
```

```
        # 在初始化模型并行之前重新应用 MindSpeed patch，
        # 因为第一次 patch 时 context_parallel_size 可能还未从 hydra 配置中获取。
        apply_patch(self.model_config, self.engine_config, self.optimizer_config)
        super()._init_device_mesh()

    def _build_megatron_module(self):
        # 判断是否为值模型（如 ForTokenClassification）
        is_value_model = (
            "ForTokenClassification" in self.model_config.architectures[0]
            or "ForSequenceClassification" in self.model_config.architectures[0]
        )
        self.is_value_model = is_value_model

        from megatron.core.enums import ModelType
        from megatron.training.training import get_model

        # 如果只做前向推理，则不需要优化器等辅助模块
        if self.engine_config.forward_only:
            module = get_model(gpt_model_provider, ModelType.encoder_or_decoder, wrap_with_
            ddp=False)
        else:
            # 训练模式，需要 DDP 包装
            module = get_model(gpt_model_provider, ModelType.encoder_or_decoder, wrap_with_
            ddp=True)

        # 加载权重（通过 vanilla bridge）
        if self.vanilla_bridge:
            self.bridge.load_weights(module, self.model_config.local_path)
        else:
            raise ValueError(f"vanilla_bridge should be true now, but got {self.vanilla_bridge}")

        # 打印模型规模（仅 rank 0）
        if torch.distributed.get_rank() == 0:
            print_model_size(module[0])

        # 如果启用 routing replay，打印相关实例数
        if self.enable_routing_replay:
            print(f"routing replay layers: {len(RouterReplay.router_instances)}")

        return module
```

verl/workers/config/engine.py

配置 dataclass 变更，重命名策略和 kwargs 字段。

```
# verl/workers/config/engine.py

@dataclass
class MindSpeedEngineConfig(McoreEngineConfig):
    """Configuration for mindspeed parallelism.
```

```
    The inheritance from BaseConfig provides omegaconf.DictConfig-like interface for a
    dataclass config.

    Args:
        mcore_kwargs dict[str, Any]: mindspeed_megatron engine kwargs.
        fsdp_kwargs dict[str, Any]: mindspeed_fsdp engine kwargs.
    """

    strategy: str = "mindspeed_megatron"
    mcore_kwargs: dict[str, Any] = field(default_factory=dict)
    fsdp_kwargs: dict[str, Any] = field(default_factory=dict)

    def __post_init__(self) -> None:
        """config validation logics go here"""
        # 验证策略仅支持 mindspeed_megatron 或 mindspeed_fsdp
        assert self.strategy in ["mindspeed_megatron", "mindspeed_fsdp"], f"strategy {self.
        strategy} not supported"
        assert self.dtype in ["bfloat16", "float16"], f"dtype {self.dtype} not supported"
        if self.tensor_model_parallel_size == 1:
            warnings.warn("set sequence parallel to false as TP size is 1", stacklevel=2)
            self.sequence_parallel = False
```

verl/workers/engine/mindspeed/utils.py

配置转换函数适配新策略和 kwargs。

```
# verl/workers/engine/mindspeed/utils.py

def get_base_mcore_config_from_engine_config(engine_config: MindSpeedEngineConfig) -> dict:
    """
    从 MindSpeedEngineConfig 创建基础 TransformerConfig 的公共部分。

    Args:
        engine_config: mindspeed engine configuration

    Returns:
        TransformerConfig 字典（不包含模型结构相关参数）
    """

    base_config = {
        # 并行配置
        "tensor_model_parallel_size": engine_config.tensor_model_parallel_size,
        "expert_model_parallel_size": engine_config.expert_model_parallel_size,
        "expert_tensor_parallel_size": engine_config.expert_tensor_parallel_size,
        "pipeline_model_parallel_size": engine_config.pipeline_model_parallel_size,
        "virtual_pipeline_model_parallel_size": engine_config.virtual_pipeline_model_parallel_size,
        "context_parallel_size": engine_config.context_parallel_size,
        "sequence_parallel": engine_config.sequence_parallel,
        "use_distributed_optimizer": engine_config.use_distributed_optimizer,
        "seed": engine_config.seed,
```

```
    }

    # 直接使用 mcore_kwargs 注入额外参数，不再按策略分支
    # 注意：当策略为 mindspeed_fsdp 时，fsdp_kwargs 不会被处理，可能需要后续扩展
    base_config.update(engine_config.mcore_kwargs)
    return base_config
```

评论区精华

与 review 评论相关的讨论：

- gemini-code-assist 评论 1：指出 verl/workers/engine/mindspeed/__init__.py 中导入未更新（但实际已更新，评论发表于代码合并前，可能基于旧版本）。
- gemini-code-assist 评论 2：建议 get_base_mcore_config_from_engine_config 中应根据策略区分 mcore_kwargs 和 fsdp_kwargs，目前代码仅使用 mcore_kwargs，当策略为 mindspeed_fsdp 时可能会丢失配置。该评论未被回应，属于未解决的设计问题。
  - Import 未及时更新导致的潜在 ImportError (correctness)：代码在最终版本中已经更新了导入，评论基于旧 diff，实际已修复。
  - FSDP 策略配置处理缺失 (design)：当前实现未处理，属于已知待改进项。

风险与影响

- 风险：
  1. 配置向后兼容性：使用旧策略名 mindspeed_llm 或旧字段 llm_kwargs/mm_kwargs 的用户配置会失效，需要迁移。
  2. FSDP 策略未完全支持：虽然配置类新增了 fsdp_kwargs，但 get_base_mcore_config_from_engine_config 未针对 mindspeed_fsdp 做特殊处理，可能导致 FSDP 运行时配置错误。
  3. 测试覆盖不完整：新增的测试脚本仅覆盖单节点 16 卡场景，多节点或超长序列场景未验证。 - 影响：用户：使用 MindSpeed 后端的用户需更新配置和启动脚本中的策略名及 kwargs；新用户可按新配置规范编写。系统：Ascend NPU 上的 GRPO 训练流程需要验证是否稳定。团队：后续若有新策略（如 mindspeed_fsdp）需补充对应配置逻辑。
- 风险标记：配置向后不兼容，FSDP 策略未完全支持，测试覆盖有限

关联脉络

- PR #6335 [megatron] chore: refactor to use Megatron-Bridge new APIs: 同样涉及 Megatron 引擎配置重构，可能与本 PR 的配置调整有重叠。