PR #7242 完整报告

verl-project/verl

[veomni] feat: add DeepSeek V4 support

合并时间：2026-08-04 21:45

原文链接：http://prhub.com.cn/verl-project/verl/pull/7242

执行摘要

- 一句话：VeOmni 引擎新增 DeepSeek V4 支持与 MXFP4 多后端回灌
- 推荐动作：值得精读。三个设计点有复用价值：一是 _build_ops_implementation_config 的「按已安装依赖字段动态过滤 + 显式设置即报错」版本解耦范式；二是 _mxfp4_staging_data 通过字节数等价的 view/dtype 重解释复用 live storage，把回灌的额外内存从「整份 MoE」降到「1/16 的 scale 缓冲」；三是回灌流程结束后用 leftover 检查兜底，任何不经过 replace_parameter 的路径都会显式失败。需要跟进的事项：为 Marlin 回灌与 converter 路径补专项测试、评估 max_num_seqs 全局默认调整、补充与 Megatron 的性能对比数据。

功能与动机

verl 此前已在 Megatron 引擎上完成 DeepSeek V4 支持（PR 7221、PR 7224），本 PR 的目标是在 VeOmni 引擎上补齐同一条能力线，让 DSv4-Flash 的大规模 GRPO/GSPO 训练可以跑在 VeOmni 上并复用其 FSDP offload、ulysses/expert 并行等能力（VeOmni 侧对应 PR 为 ByteDance-Seed/VeOmni#962）。PR body 中贴出的 rollout_corr 指标（rollout_actor_probs_pearson_corr 0.9925、rollout_corr/kl 0.0053、ppl_ratio 1.0045）用于证明训练端与 rollout 端概率分布一致；同时预告了后续两步：R3 hash router replay 与 F8/FP4 QAT 训练。

实现拆解

实现按 5 步拆解：

1. 引擎装配与模型构建（verl/workers/engine/veomni/transformer_impl.py） - 新增 _build_ops_implementation_config：以已安装 VeOmni 的 OpsImplementationConfig 字段集合为准，动态过滤 verl 侧全部 *_implementation 配置；未知字段若处于 verl 默认值则跳过并记日志，若被用户显式设置则抛 ValueError。这是为了处理 verl 构建版本新于已安装 VeOmni 时构造器报未知关键字的问题，同时保证用户点名的 kernel 不会被静默降级。 - _build_model_optimizer 的 build_parallelize_model 调用新增 broadcast_model_weights_from_rank0=True 与 fqn_to_index_mapping=load_safetensors_index(...)，配合 veomni/utils.py 新增的 load_safetensors_index 读取 model.safetensors.index.json，把每个 FQN 映射到分片编号，支撑 rank0 广播加载，避免大规模模型每 rank 全量读权重文件。 - get_per_tensor_param 新增 DSv4 快捷路径：若 VeOmni 的

get_checkpoint_tensor_converter 返回的 converter 具备 export_weights，直接用 converter 导出权重并返回 (weights, None)；代码注释自述该路径当前仅限 DeepseekV4，其余模型仍走 state_dict + param_generator 旧路径。旧路径同时修复了 CPU offload 场景：DTensor 需先 .to(device, non_blocking=True) 再 full_tensor()，避免整模半迁移导致 state_dict() 崩溃。

2. 配置扩展（verl/workers/config/engine.py、各 generated YAML） - VeOmniEngineConfig 新增 dsa_indexer_implementation、dsa_attention_implementation、mhc_implementation 三个 kernel 选择器，默认 eager；verl/trainer/config/engine/veomni.yaml 与 _generated_ppo_veomni_trainer.yaml 三处（actor/ref/critic）同步补齐。 - 所有 generated 配置（ppo/veomni/megatron/torchtitan 及 rollout.yaml）将 vLLM rollout 的 max_num_seqs 默认值从 1024 调为 256，这是一个超出 DSv4 范围的全局默认变更。
3. vLLM MXFP4 MoE 回灌泛化（verl/utils/vllm/vllm_fp4_utils.py） - 用 _refittable_mxfp4_backends 取代原先的 _is_deepgemm_mxfp4_moe_module 判断，支持后端从仅 DeepGEMM 扩到 DeepGEMM/MARLIN/BATCHED_MARLIN；Triton 明确排除（权重是非 Parameter 包装，无法被 _replace_parameter_in_place 拦截）。 - _mxfp4_checkpoint_layout 追加 w13_bias/w2_bias 条目：无条件列出、缺失时跳过，防止 Marlin 的 bias 置换落入真实 replace_parameter 而悄悄搬走缓冲区。 - 新增 _mxfp4_staging_data：字节数等价的场景（Marlin int32 tile 与 uint8 checkpoint 权重同字节数）用 view + dtype 重解释复用 live storage，回灌额外内存从整份 MoE 降到 1/16 的 scale 缓冲；否则用 zeros 兜底。 - stage_mxfp4_moe_params_for_loading 对不支持的后端改为显式 NotImplementedError，错误信息列出支持列表。
4. 权重同步与显存处理（verl/workers/engine_workers.py） - naive 权重同步路径在 set_expandable_segments(False) 之后追加 aggressive_empty_cache(force_sync=True)，在 rollout resume 前同步清空显存缓存，为超大模型权重唤醒腾出确定的内存。
5. 示例与测试配套 - 新增 examples/grpo_trainer/run_deepseek_v4_veomni.sh：8 卡 FSDP offload + ulysses/expert 并行，MoE 用 fused_triton，DSA indexer/attention 与 MHC 用 tilelang，rollout 侧 vLLM（TP=1、EP=8、kv_cache_dtype=fp8），算法侧为 GSPO。 - run_deepseek_v4_flash_megatron.sh 删除 1 行，结合最后一个 commit「remove deep_gemm config」推测为移除 deep_gemm 相关配置行（diff 未展示具体内容，属推断）。 - tests/workers/test_router_replay_engine_helpers_on_cpu.py 仅为新增 import 补充 sys.modules mock；Marlin 回灌与 converter 路径都没有新增专项测试。

关键文件：

- verl/utils/vllm/vllm_fp4_utils.py（模块 量化回灌；类别 source；类型 core-logic；符号 _refittable_mxfp4_backends, _mxfp4_staging_data, _mxfp4_checkpoint_layout, stage_mxfp4_moe_params_for_loading）：MXFP4 MoE 回灌的核心泛化：从仅 DeepGEMM 扩展到 DeepGEMM/Marlin/BatchedMarlin，用字节级存储重解释避免回灌时 MoE 内存翻倍，并新增 bias 条目防止 Marlin 置换悄悄搬走缓冲区。
- verl/workers/engine/veomni/transformer_impl.py（模块 引擎实现；类别 source；类型 dependency-wiring；符号 _build_ops_implementation_config, get_per_tensor_param, _build_model_optimizer）：VeOmni 引擎装配的核心改动：新增版本感知的 ops config 构

造器、rank0 广播加载与 DSv4 converter 权重导出路径，并修复 CPU offload 下的 DTensor 处理。

- verl/workers/engine/veomni/utils.py（模块 权重加载；类别 source；类型 core-logic；符号 load_safetensors_index）：新增 load_safetensors_index，为 rank0 广播加载提供 FQN 到分片编号的映射，是超大模型加载路径的地基。
- verl/workers/config/engine.py（模块 引擎配置；类别 source；类型 core-logic；符号 dsa_indexer_implementation, dsa_attention_implementation, mhc_implementation）：VeOmniEngineConfig 新增 DSA/MHC 三个 kernel 选择器配置项，是示例脚本中 tilelang 内核选择的配置入口。
- examples/grpo_trainer/run_deepseek_v4_veomni.sh（模块 示例脚本；类别 other；类型 configuration）：新增的 DSv4 GRPO 训练示例，完整展示了 VeOmni 引擎配置（FSDP offload、ulysses/expert 并行、tilelang DSA/MHC）与 vLLM rollout 搭配方式。
- verl/workers/engine_workers.py（模块 权重同步；类别 source；类型 core-logic；符号 update_weights）：naive 权重同步路径追加 aggressive_empty_cache(force_sync=True)，在 rollout resume 前强制同步清空显存，属超大模型场景的关键内存处理。
- verl/trainer/config/_generated_ppo_veomni_trainer.yaml（模块 训练配置；类别 config；类型 configuration）：VeOmni 默认训练配置生成文件，同步新增 DSA/MHC 配置项并将 rollout max_num_seqs 默认值从 1024 调为 256。
- tests/workers/test_router_replay_engine_helpers_on_cpu.py（模块 单元测试；类别 test；类型 test-coverage）：唯一改动的测试文件，仅为新增 VeOmni import 补充 sys.modules mock；反映本 PR 对新路径缺乏专项测试覆盖。
- verl/trainer/config/engine/veomni.yaml（模块 引擎配置；类别 config；类型 configuration）：VeOmni 引擎配置模板，同步新增 DSA/MHC 三个 kernel 选择器默认值。
- verl/trainer/config/_generated_ppo_megatron_trainer.yaml（模块 训练配置；类别 config；类型 configuration）：Megatron 默认配置同样被 max_num_seqs 全局调整波及，说明该默认值变更影响所有引擎。
- verl/trainer/config/_generated_ppo_torchtitan_trainer.yaml（模块 训练配置；类别 config；类型 configuration）：Torchtitan 默认配置被 max_num_seqs 全局调整波及。
- verl/trainer/config/_generated_ppo_trainer.yaml（模块 训练配置；类别 config；类型 configuration）：通用 PPO 默认配置被 max_num_seqs 全局调整波及。
- verl/trainer/config/rollout/rollout.yaml（模块 训练配置；类别 config；类型 configuration）：rollout 配置模板的 max_num_seqs 默认值调整，是全局默认并发上限变更的源头之一。
- examples/grpo_trainer/run_deepseek_v4_flash_megatron.sh（模块 示例脚本；类别 other；类型 configuration）：Megatron 侧 DSv4 示例删除 1 行配置，与最后一个 commit「remove deep_gemm config」呼应，体现后端支持口径的统一。

关键符号：_build_ops_implementation_config, load_safetensors_index, _refittable_mxfp4_backends, _mxfp4_staging_data, _mxfp4_checkpoint_layout, stage_mxfp4_moe_params_for_loading, _stage_mxfp4_moe_params, _process_mxfp4_moe_params, get_per_tensor_param, update_weights

关键源码片段

verl/utils/vllm/vllm_fp4_utils.py

MXFP4 MoE 回灌的核心泛化：从仅 DeepGEMM 扩展到 DeepGEMM/Marlin/BatchedMarlin，用字节级存储重解释避免回灌时 MoE 内存翻倍，并新增 bias 条目防止 Marlin 置换悄悄搬走缓冲区。

```
# verl/utils/vllm/vllm_fp4_utils.py
# MXFP4 MoE 回灌（refit）的后端白名单：转换必须是以 checkpoint layout 为输入的纯函数，
# 并且结果必须经由 replace_parameter 发布，_replace_parameter_in_place 才能拦截并
# 把新权重写回 CUDA graph 已捕获的 live storage。
# Triton 是明确排除项：它把权重写成非 Parameter 包装、scale 挂在 quant_method 上，
# kernel 读取的缓冲区根本不在 Parameter 体系里，无法被拦截。

def _refittable_mxfp4_backends():
    from vllm.model_executor.layers.fused_moe.oracle.mxfp4 import Mxfp4MoeBackend

    return (
        Mxfp4MoeBackend.DEEPGEMM_MXFP4,
        Mxfp4MoeBackend.MARLIN,
        Mxfp4MoeBackend.BATCHED_MARLIN,
    )


def _mxfp4_staging_data(param, shape, dtype):
    """为 load_weights 构造 checkpoint layout 的暂存张量。

    只要字节数相同就复用 live storage：Marlin 的 repack 本质是 4bit 值的置换，
    其 int32 tile 与 uint8 checkpoint 权重字节数完全一致，view + dtype 重解释后
    reload 直接落在 kernel 读取的地址上，回灌的额外内存从整份 MoE（近乎翻倍）
    降到只有 block scale 的 1/16 缓冲。
    """
    data = param.data
    shape = torch.Size(shape)
    if data.dtype == dtype and data.shape == shape:
        # 形状 dtype 完全一致，直接原地吸收
        return data
    if data.is_contiguous() and data.numel() * data.element_size() == shape.numel() * dtype.
    itemsize:
        # 字节数等价：flatten 后按目标 dtype 重新解释并 reshape，零拷贝
        return data.flatten().view(dtype).reshape(shape)
    # zeros 而不是 empty：refit 流若跳过某参数，零填充会让该层输出坍缩为 0，
    # 而不是让 kernel 读取释放后内存里的随机数据。
    return torch.zeros(shape, dtype=dtype, device=param.device)


# _mxfp4_checkpoint_layout 中新增的 bias 条目是无条件列出的：
# 无 MoE bias 的模型从未注册这些参数，stage 时遇到缺失会自动跳过；
```

```
# 但若不列出，Marlin 的 bias 置换会落进真实的 replace_parameter，
# 悄悄搬走缓冲区且不经过 leftover 检查。

def stage_mxfp4_moe_params_for_loading(model):
    """给 load_weights 发 checkpoint layout 的 expert 缓冲，返回待处理模块列表。"""
    supported = _refittable_mxfp4_backends()
    staged_modules = []
    backends = set()
    for module in model.modules():
        if not _is_mxfp4_fused_moe_module(module):
            continue
        backend = getattr(module.quant_method, "mxfp4_backend", None)
        if backend not in supported:
            # 显式拒绝胜过静默跳过：不支持的 backend 会把 checkpoint layout 数据
            # 写进已被重排的参数里，rollout 概率漂移且没有任何报错指向原因。
            raise NotImplementedError(
                f"mxfp4 MoE refit does not support the {backend} backend selected by "
                f"{type(module).__name__}. Supported backends: "
                f"{', '.join(str(b) for b in supported)}."
            )
        _stage_mxfp4_moe_params(module)
        staged_modules.append(module)
        backends.add(backend)

    logger.info(
        "Staged %d mxfp4 MoE modules for in-place refit (backends: %s)",
        len(staged_modules),
        ", ".join(sorted(str(b) for b in backends)) or "none",
    )
    return staged_modules
```

verl/workers/engine/veomni/transformer_impl.py

VeOmni 引擎装配的核心改动：新增版本感知的 ops config 构造器、rank0 广播加载与 DSv4 converter 权重导出路径，并修复 CPU offload 下的 DTensor 处理。

```
# verl/workers/engine/veomni/transformer_impl.py
# verl 的 VeOmniEngineConfig 与 VeOmni 的 OpsImplementationConfig 逐字段镜像。
# 当 verl 构建版本新于已安装的 VeOmni 时，直接构造会因未知关键字报错；
# 这里按“已安装 VeOmni 实际接受的字段”动态过滤，避免 kernel 选择被静默降级。

def _build_ops_implementation_config(engine_config: VeOmniEngineConfig) ->
OpsImplementationConfig:
    # 以已安装 VeOmni 的字段集合为准，而不是以 verl 侧配置为准
    accepted = {f.name for f in fields(OpsImplementationConfig)}

    kwargs, unsupported, skipped = {}, {}, []
    for f in fields(engine_config):
        if not f.name.endswith("_implementation"):
            continue
```

```
        value = getattr(engine_config, f.name)
        if f.name in accepted:
            kwargs[f.name] = value
        elif value != f.default:
            # 用户显式点名的 kernel 在旧版 VeOmni 上不存在，直接报错而不是降级
            unsupported[f.name] = value
        else:
            skipped.append(f.name)

    if unsupported:
        raise ValueError(
            f"The installed VeOmni's OpsImplementationConfig has no {sorted(unsupported)}, but "
            f"they were explicitly set to {unsupported}. Upgrade VeOmni or unset these options."
        )
    if skipped:
        logger.info(
            f"Skipping {sorted(skipped)}: unknown to the installed VeOmni, left at the verl default."
        )

    return OpsImplementationConfig(**kwargs)


# _build_model_optimizer 中：并行化时传入 safetensors 分片索引并允许 rank0
# 广播权重复本，避免每个 rank 都全量加载大模型权重文件（DSv4 类千亿模型关键）。
module = build_parallelize_model(
    module,
    init_device=self.engine_config.init_device,
    weights_path=self.model_config.local_path,
    enable_full_shard=self.engine_config.enable_full_shard,
    mixed_precision=veomni_mixed_precision_config,
    enable_gradient_checkpointing=self.model_config.enable_gradient_checkpointing,
    enable_fsdp_offload=self.engine_config.enable_fsdp_offload,
    basic_modules=list(
        set(getattr(module, "_no_split_modules", None) or []) | set(self.engine_config.basic_
        modules)
    ),
    enable_reentrant=self.engine_config.enable_reentrant,
    enable_forward_prefetch=self.engine_config.forward_prefetch,
    broadcast_model_weights_from_rank0=True,
    fqn_to_index_mapping=load_safetensors_index(self.model_config.local_path),
)


# get_per_tensor_param 中：DSv4 走 VeOmni 的 checkpoint converter 直接导出权重，
# 返回 (weights, None) 表示无 peft 配置；其余模型仍走 state_dict 旧路径。
# 注意：新建路径当前仅限 DeepseekV4，后续计划统一所有模型（代码内已留 TODO）。
converter = get_checkpoint_tensor_converter(self.module)
if converter is not None and hasattr(converter, "export_weights"):
    return converter.export_weights(self.module), None
```

verl/workers/engine/veomni/utils.py

新增 load_safetensors_index，为 rank0 广播加载提供 FQN 到分片编号的映射，是超大模型加载路径的地基。

```
# verl/workers/engine/veomni/utils.py
# 读取 HF safetensors 分片索引，把每个 FQN 映射到其所在分片文件的编号，
# 供 build_parallelize_model 的 fqn_to_index_mapping 参数使用，
# 配合 broadcast_model_weights_from_rank0 实现 rank0 广播按需加载。

def load_safetensors_index(model_path: str) -> dict[str, int]:
    path = os.path.join(model_path, "model.safetensors.index.json")
    if not os.path.exists(path):
        # 非分片权重（单文件或无索引）时返回空映射，调用方按旧逻辑处理
        return {}
    with open(path) as f:
        weight_map = json.load(f)["weight_map"]
    # 文件名形如 model-00001-of-00010.safetensors，取中间的分片编号
    index = {fqn: int(filename.split("-")[1]) for fqn, filename in weight_map.items()}
    return index
```

评论区精华

PR 全程没有 review 评论，唯一的互动是 Issue 评论区 vermouth1992 的提问「How does the speed compare with Megatron?」，作者未回复。该问题指向 VeOmni 与 Megatron 双引擎的取舍：本 PR 只提供了训练 /rollout 概率一致性指标（actor_probs_pearson_corr 0.9925、kl 0.0053、ppl_ratio 1.0045）作为正确性证据，性能对比仍缺失。决策结论：功能按原设计合入（作者自行合入），性能问题留待后续验证。

- VeOmni 与 Megatron 引擎的性能对比 (question): 作者未回复；PR 内只提供了训练 /rollout 概率一致性指标（pearson 0.9925、kl 0.0053、ppl_ratio 1.0045）作为正确性证据，没有与 Megatron 的吞吐对比基准。

风险与影响

- 风险：
  1. 测试缺口：MXFP4 Marlin/BatchedMarlin 回灌和 VeOmni DSv4 converter 加载路径没有任何专项测试，tests/workers/test_router_replay_engine_helpers_on_cpu.py 只是为新增 import 补 sys.modules mock；这些逻辑深度依赖 vLLM 内部 replace_parameter、weight_loader 与 VeOmni checkpoint converter 的行为契约，上游升级可能悄悄破坏。
  2. 双路径并存：get_per_tensor_param 中 converter 导出与 state_dict/param_generator 旧路径并存，代码注释自认「currently only for DeepseekV4」，后续需统一，期间两条路径都要维护。
  3. 全局默认变更：所有 generated 配置（veomni/megatron/torchtitan/ppo 与 rollout.yaml）的 max_num_seqs 从 1024 改为 256，影响所有引擎的默认 rollout 并发，可能降低小模型默认吞吐，需要确认是否有意为之。

4. 性能未验证：aggressive_empty_cache(force_sync=True) 追加在 naive 权重同步关键路径上，引入同步清缓存开销；与 Megatron 的对比问题未答复，性能风险未量化。
5. 版本耦合：dsa_*/mhc_implementation 三个新配置依赖 VeOmni 0.1.12+，老版本显式设置会抛 ValueError，用户需先升级 VeOmni 才能显式使用这些 kernel。 - 影响：用户侧：DeepSeek V4-Flash 用户新增一条 VeOmni 训练路径（FSDP offload + ulysses/expert 并行 + tilelang DSA/MHC，配合 vLLM rollout 与 kv_cache_dtype=fp8）；同时 vLLM 量化 MoE 回灌从 DeepGEMM 扩展到 Marlin/BatchedMarlin，Hopper 用户直接受益。系统侧：DSv4 支持从 Megatron 单引擎扩展到 VeOmni 双引擎，为后续 R3 hash router replay 与 FP8/FP4 QAT 训练铺路。流程侧：作者自合且无 review，11 次提交中可见多轮返工（fqn_to_index_mapping 提交两次、DSA 配置、CPU offload 修复、CI 修复等），说明与 VeOmni 上游 PR 962 的联调过程较曲折。 - 风险标记：缺少专项测试，全局默认配置变更，双导出路径并存，性能对比待验证，自合并且无 review

关联脉络

- PR #7224 [vllm] feat: enhance DeepSeek V4 fp8/fp4 linear and moe weight refit: 同属 DeepSeek V4 支持线，聚焦 vLLM FP8/FP4/MXFP4 权重 refit；本 PR 对 vllm_fp4_utils.py 的泛化（Marlin + bias 条目）与其直接衔接，且两边都改了 run_deepseek_v4_flash_megatron.sh。
- PR #7221 [megatron] feat: support contiguous context-parallel layout for DeepSeek V4: Megatron 侧为 DeepSeek V4 增加 contiguous CP 布局与 THD 支持；本 PR 是 VeOmni 侧的对应实现，两条引擎路径共同构成 verl 对 DSv4 的训练能力。
- PR #7243 [veomni] fix: preserve GPT-OSS weights without expert parallelism: 紧随本 PR 的 veomni 修复，落在相同的 veomni/utils.py 与 transformer_impl.py 上修正 GPT-OSS 权重导出问题，说明本 PR 引入的 exporter/ 权重导出路径仍在持续打磨。