PR #7272 完整报告

verl-project/verl

[fsdp,veomni] feat: support pad_to_length to reduce jit compile time

合并时间：2026-08-06 09:01

原文链接：http://prhub.com.cn/verl-project/verl/pull/7272

执行摘要

- 一句话：FSDP/VeOmni 新增 pad_to_length 减少 JIT 重复编译
- 推荐动作：值得精读的 perf 型 PR。重点关注三处设计：一是"按 attention workload 均衡分 micro-batch 导致 token 数越界，因此必须用细 bucket 而非固定预算对齐"的量化论证；二是 prepare_model_inputs 中补位时机（roll 之后、SP 切分之前）与 _gather_and_unpad_packed 统一裁剪的配合；三是 router replay 在补位下的掩码处理——补位 token 无 RECORD 记录所以必须 gate 掉走原生路由，而真实 token 打 gate 才会破坏 R2 bit-equal 保证。测试文件对这两个引擎用 MagicMock 桩掉 veomni.* 依赖的做法也值得复用。

功能与动机

PR body 明确指出痛点：With use_remove_padding=True + use_dynamic_bsz=True, every packed micro-batch has a different token count, so shape-specialized kernels (torch.compile, Triton/DeepGEMM autotune) recompile or re-autotune continuously throughout training。并且解释了为什么必须用 bucket 对齐而非固定目标长度：rearrange_micro_batches 按 token 预算推导微批数量，却按 attention 负载（Σ 24576·seqlen + seqlen²）做 Karmarkar-Karp 均衡分配样本，导致微批 token 数既非恒定也不受预算约束，直接对齐预算会留下大量"完全动态"的超预算微批。

实现拆解

实现按 5 步展开，全部围绕「让 packed 微批形状稳定」这一目标：

1. 新增张量级补位工具 pad_packed_inputs（verl/workers/engine/utils.py）：对 (1, total_nnz) 的 packed 序列右补 pad_size 个 token。关键约定是追加的 position_ids 从 0 重新开始，使补位 token 在 cu_seqlens 中自成一个尾部 varlen 段，而非把最后一条真实序列拉长；同时兼容 mRoPE 的 (rope_dim, 1, total_nnz) 三维布局，并支持自定义 pad_value（temperature 补位必须用 1 而不是 0，因为它会除进 logits）。
2. FSDP 引擎接入（verl/workers/engine/fsdp/transformer_impl.py）：FSDPEngine.__init__ 从配置读取 pad_to_length / pad_to_length_bucket；新增 _get_packed_pad_size 用 ceildiv 上取整到 bucket 倍数；prepare_model_inputs 在 torch.roll 之后、SP 切分之前计算 static_pad_size 并对 input_ids_rmpad、position_ids_rmpad、input_ids_rmpad_rolled、temperature_rmpad 统一补位，随后把 output_args["pad_size"] 从"仅 SP 对齐"扩展为"静态 + SP 对齐"的总和；

prepare_model_outputs 中原来散落的 if self.use_ulysses_sp: gather_outputs_and_unpad(...) 分支统一收敛到新增的 _gather_and_unpad_packed，保证 log_probs、entropy、sum_pi_squared、distillation 辅助量全部按总 pad 裁剪。

3. VeOmni 引擎与 router replay 适配（verl/workers/engine/veomni/transformer_impl.py）：VeOmniEngineWithLMHead 同样读取 pad 配置（继承 _get_packed_pad_size）；_maybe_push_router_replay_state 在 REPLAY 分支先记录 real_nnz = flat.size(0)，再按 output_args["pad_size"] 把路由目标 flat 扩展 pad_size 行、把掩码尾部置 False 后重新走 slice_microbatch_replay_targets / slice_microbatch_replay_mask。这里的正确性论证是：补位 token 没有 RECORD 记录，必须被 gate 掉走原生路由，而对真实 token 打 gate 才会破坏 R2 的 bit-equal 保证。
4. 配置与入口联动：verl/workers/config/engine.py 给 FSDPEngineConfig 和 VeOmniEngineConfig 各加 pad_to_length: bool = False、pad_to_length_bucket: int = 1024 两个字段及详细 docstring；verl/workers/config/actor.py 为 FSDPActorConfig、VeOmniActorConfig 增加转发字段；同步更新 fsdp.yaml、veomni.yaml、veomni_actor.yaml、dp_actor.yaml、dp_ref.yaml、veomni_ref.yaml 和两个 _generated_*.yaml；run_deepseek_v4_veomni.sh、run_deepseek_v4_flash_megatron.sh 两个示例脚本按新配置路径修正参数。
5. 测试配套：新增 tests/workers/test_packed_pad_to_length_on_cpu.py（186 行），用 MagicMock 桩掉 veomni.* 模块，覆盖 pad_size=0 无操作、补位自成一 varlen 段、mRoPE 逐 rope_dim 补位、pad 值与 dtype 保持、bucket 取整性质、超预算微批仍落 bucket、以及 top-K 蒸馏守卫；test_router_replay_engine_helpers_on_cpu.py 新增 R2/R3 在 pad_to_length 下的 2 个测试；test_distillation_topk_symmetry_on_cpu.py 增加 2 行守卫断言。

关键文件：

- verl/workers/engine/fsdp/transformer_impl.py（模块 FSDP 引擎；类别 source；类型 core-logic；符号 _get_packed_pad_size, _gather_and_unpad_packed, prepare_model_inputs, prepare_model_outputs）：核心实现所在：新增 _get_packed_pad_size / _gather_and_unpad_packed，prepare_model_inputs 完成静态补位，prepare_model_outputs 统一按总 pad 裁剪，并加 top-K 蒸馏守卫。
- verl/workers/engine/utils.py（模块 引擎工具；类别 source；类型 core-logic；符号 pad_packed_inputs）：新增 pad_packed_inputs 核心工具函数，定义补位 token 的 position_ids 约定（自成 varlen 段）与 mRoPE 兼容逻辑。
- verl/workers/engine/veomni/transformer_impl.py（模块 VeOmni 引擎；类别 source；类型 core-logic；符号 _maybe_push_router_replay_state, VeOmniEngineWithLMHead）：VeOmni 引擎接入同一配置，并完成 router replay（R2/R3）在补位下的目标扩展与掩码 gate，是本 PR 后半部分正确性核心。
- tests/workers/test_packed_pad_to_length_on_cpu.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_engine, TestPadPackedInputs, test_pad_size_zero_is_a_noop, test_pad_tokens_form_their_own_varlen_segment）：新增 186 行 CPU 单测，覆盖补位工具、bucket 取整性质、超预算微批、蒸馏守卫与解补裁剪，并用 MagicMock 桩掉 veomni 依赖。

- verl/workers/config/engine.py（模块 引擎配置；类别 source；类型 configuration；符号 pad_to_length, pad_to_length_bucket）：FSDPEngineConfig 与 VeOmniEngineConfig 新增 pad_to_length / pad_to_length_bucket 字段与详尽文档，是整个特性的配置契约入口。
- tests/workers/test_router_replay_engine_helpers_on_cpu.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_r2_pad_to_length_extends_targets_and_gates_the_pad_tail, test_r3_pad_to_length_gates_both_prompt_and_pad）：新增 R2/R3 在 pad_to_length 下目标扩展与补位尾部 gate 的测试，锁住 router replay 与补位的交互契约。

关键符号：pad_packed_inputs, _get_packed_pad_size, _gather_and_unpad_packed, prepare_model_inputs, prepare_model_outputs, _maybe_push_router_replay_state

关键源码片段

verl/workers/engine/utils.py

新增 pad_packed_inputs 核心工具函数，定义补位 token 的 position_ids 约定（自成 varlen 段）与 mRoPE 兼容逻辑。

```
# verl/workers/engine/utils.py
def pad_packed_inputs(
    input_ids_rmpad: torch.Tensor,
    position_ids_rmpad: torch.Tensor | None,
    pad_size: int,
    pad_value: float = 0,
):
    """右补 packed 序列 (1, total_nnz) 尾部 pad_size 个 token。

    与 verl.utils.ulysses.ulysses_pad 补到序列并行倍数的方式保持一致：
    追加的 position_ids 从 0 重新开始，因此补位 token 在 cu_seqlens 里
    自成一个尾部 varlen 段，而不是把最后一条真实序列"拉长"。
    """
    if pad_size <= 0:
        return input_ids_rmpad, position_ids_rmpad

    input_ids_rmpad = torch.nn.functional.pad(input_ids_rmpad, (0, pad_size), value=pad_value)
    if position_ids_rmpad is not None:
        # 单段 position_ids 形状为 (1, pad_size)；mRoPE 下为 (rope_dim, 1, pad_size)
        pad_position_ids = torch.arange(
            pad_size, dtype=position_ids_rmpad.dtype, device=position_ids_rmpad.device
        ).unsqueeze(0)
        if position_ids_rmpad.dim() == 3: # (rope_dim, 1, total_nnz) mRoPE 布局
            pad_position_ids = pad_position_ids.unsqueeze(0).repeat(position_ids_rmpad.size(0), 1,
            1)
        position_ids_rmpad = torch.cat((position_ids_rmpad, pad_position_ids), dim=-1)
    return input_ids_rmpad, position_ids_rmpad
```

verl/workers/engine/veomni/transformer_impl.py

VeOmni 引擎接入同一配置，并完成 router replay（R2/R3）在补位下的目标扩展与掩码 gate，是本 PR 后半部分正确性核心。

```
# verl/workers/engine/veomni/transformer_impl.py
# _maybe_push_router_replay_state 的 REPLAY 分支（节选）
# flat 是 [mb_nnz, L, topk] 的压平路由目标；pad_to_length 会往 input_ids
# 尾部追加补位 token，这些位置在 RECORD 阶段没有记录路由，因此要把目标
# 扩展到补位后的长度，并给尾部打 False gate 让它们走原生路由。
# 对真实 token 打 gate 才会破坏 R2 的 bit-equal 前向保证，而补位 token
# 的输出在进入 loss 前就被丢弃，gate 掉是安全的。
real_nnz = flat.size(0)

# R3：response 位置才有真实记录，prompt 位置是占位零值，需先构建逐 token
# 的 mask_flat（response_mask 相关校验与构造逻辑此处省略）
mask_flat = None
if self._router_replay_mode == "R3":
    ...

pad_size = int(output_args.get("pad_size", 0))
if pad_size:
    if mask_flat is None:
        # R2 默认全 True：真实 token 全部回放，保证 bit-equal 前向
        mask_flat = torch.ones(real_nnz, dtype=torch.bool, device=flat.device)
    mask_flat = torch.cat([mask_flat, mask_flat.new_zeros(pad_size)])
    flat = torch.cat([flat, flat.new_zeros((pad_size, *flat.shape[1:]))])

per_layer = rr.slice_microbatch_replay_targets(flat)
# 与 routed_experts 使用同一套 pad + slice 规则切分 mask
replay_mask = rr.slice_microbatch_replay_mask(mask_flat) if mask_flat is not None else None
rr.set_microbatch_targets(per_layer, replay_mask=replay_mask)
```

评论区精华

该 PR 无任何 review 评论，reviewer Luosuu 直接 APPROVED。真正的设计讨论沉淀在 PR body：作者用一组量化数据对比三种补位方案——“对齐预算、超预算保持动态”会带来 +24.8% token 开销且形状无界；“上取整到预算倍数”形状收敛但开销高达 +49.5%；而“按 1024 bucket 对齐”（本 PR 方案）仅 +1.7% 开销且形状收敛到 {6144, 10240, 12288}。另外 4 个 commit（[fsdp,veomni] feat → simplify → fix ci → pad to bucket）显示 bucket 对齐是最后一轮才定型的方向，前两轮是更简单的固定目标补位，后来才针对 rearrange_micro_batches 的 workload 均衡特性改成细粒度 bucket。

- bucket 对齐 vs 固定目标长度对齐的设计权衡 (design): 采用细粒度 bucket 方案；提交历史中 'pad to bucket' 是最后一个 commit，说明该方案在迭代末期定型。无 reviewer 异议，Luosuu 直接 approve。
- router replay 在补位下如何保证正确性 (correctness): 实现中对 real token 保持原掩码语义，仅对补位尾部追加 False gate，并断言 flat 扩展前后长度一致；新增 R2/R3 两个测试锁住该行为。

风险与影响

- 风险：具体风险点如下：
  - 核心训练路径变更：prepare_model_inputs / prepare_model_outputs 是 FSDP/VeOmni 共用热路径，output_args["pad_size"] 语义从“仅 SP 对齐”扩展为“SP + 静态”总和，任何仍只读 SP 部分的消费方都会漏裁导致序列错位。代码通过收敛到 _gather_and_unpad_packed 单点裁剪来降低此风险。
  - 补位 token 的隐形开销：补位 token 仍走完整前向（含 EP all-to-all），bucket 太细则形状集合大、收益下降，太粗则浪费算力；默认 1024 是经验值，文档建议按 max_token_len_per_gpu * sp 配置。
  - 补位 token 的语义边界：input_ids 补位值默认 0 可能与真实 token id 0 相同，但因补位 token 自成 varlen 段且输出在进入 loss 前被丢弃，不会污染训练。
  - router replay 交互：R2 此前从不带 mask，现在会带“真实 token 全 True + 补位尾部 False”的掩码，pad 尾部走原生路由会略微改变 EP 负载分布；控制器 set_microbatch_targets 的掩码语义已同步放宽并加测试锁住。
  - 功能边界：top-K distillation 与 pad_to_length 不兼容（teacher 张量只按 Ulysses 规则切分、不知晓静态补位），已用 RuntimeError 显式拒绝，避免静默错位。
  - 影响：对用户：开启 pad_to_length=true 后，动态 batch 训练的 packed 长度收敛到有限形状集合，torch.compile / Triton / DeepGEMM 的编译与 autotune 从“每步一次”变为“训练早期一次”，以约 +1.7% token 前向开销换取大幅减少的 JIT 卡顿；默认关闭，未开启用户无任何行为变化。对系统：改动覆盖 FSDP 与 VeOmni 两个引擎的输入输出管线及配置契约，新增 2 个配置键并同步到全部生成配置；VeOmni router replay 的 R2/R3 掩码逻辑是本 PR 后半部分新增的复杂度。对团队：后续若把该特性推广到 Torchtitan/Megatron 引擎，需要复制同样的补位与裁剪管线；pad_to_length 已成为 verl 侧的引擎级配置，与 VeOmni 上游 train.pad_to_length 语义对齐。
  - 风险标记：核心训练路径变更，补位 token 增加前向开销，Ulysses SP 补位语义合并，router replay 掩码交互

关联脉络

- PR #7242 [veomni] feat: add DeepSeek V4 support: 同样改动 verl/workers/engine/veomni/transformer_impl.py 与示例 run_deepseek_v4_veomni.sh；本 PR 是 verl 侧对 VeOmni 上游 train.pad_to_length 的引擎级移植，DeepSeek V4 VeOmni 示例是主要采纳场景。
- PR #7224 [vllm] feat: enhance DeepSeek V4 fp8/fp4 linear and moe weight refit: 同样触碰 run_deepseek_v4_flash_megatron.sh；两个 PR 都把 DeepSeek V4 示例脚本作为配置契约的联调面，说明该示例线是 pad_to_length 等新配置的主要落地路径。
- PR #7225 [algo] fix: micro-batch normalization for distillation loss: 蒸馏损失功能领域相关：本 PR 显式拒绝 pad_to_length 与 top-K distillation 组合（RuntimeError 守卫），与蒸馏相关的前向布局改动同属需保持张量流对齐的区域。