PR #6446 完整报告

verl-project/verl

[megatron] feat: Support Megatron chunk entropy

合并时间：2026-06-18 10:40

原文链接：http://prhub.com.cn/verl-project/verl/pull/6446

执行摘要

- 一句话：在 Megatron 后端实现分块熵计算以降低显存峰值
- 推荐动作：值得精读，特别是分块熵在张量并行环境下的实现方式，以及如何在多个 engine 后端统一扩展配置。关注其通过配置开关优雅降级的设计，以及 review 中关于 dtype 和参数传递的问题修复。

功能与动机

目前 chunk entropy 仅在 FSDP 后端可用，Megatron 尚未实现。该功能可显著降低长序列场景下的峰值 GPU 内存，例如在 Qwen3.5-35B Megatron 测试中，序列长度从 2k 扩展到 20k 时峰值内存减少 8.6GB。

实现拆解

1. 核心熵分块函数：在 verl/utils/megatron/tensor_parallel.py 中新增 vocab_parallel_entropy_with_chunking 函数，通过沿序列维度分块调用 _VocabParallelEntropy 自定义算子，降低单次计算的内存峰值。默认分块大小为 2048。
2. 配置扩展：在 verl/workers/config/engine.py 中的 McoreEngineConfig、FSDPEngineConfig、VeOmniEngineConfig、TorchtitanEngineConfig、AutomodelEngineConfig 类中添加 entropy_from_logits_with_chunking 布尔标志和 entropy_from_logits_chunk_size 整型配置，并同步更新 verl/workers/config/actor.py 中的对应配置类。
3. Engine 集成：在 verl/workers/engine/megatron/transformer_impl.py、verl/workers/engine/fsdp/transformer_impl.py、verl/workers/engine/automodel/transformer_impl.py、verl/workers/engine/torchtitan/transformer_impl.py 的 logits_processor 或 prepare_model_outputs 方法中，根据 entropy_from_logits_with_chunking 标志选择带分块参数的调用路径，保持向后兼容。
4. 配置文件同步：更新了 verl/trainer/config/ 下的多个 YAML 配置文件（dp_actor.yaml、_generated_ppo_trainer.yaml、ref/dp_ref.yaml、engine/megatron.yaml、_generated_ppo_megatron_trainer.yaml），将新配置项添加到默认配置中。

关键文件：

- verl/utils/megatron/tensor_parallel.py（模块 工具函数；类别 source；类型 core-logic；符号 vocab_parallel_entropy_with_chunking）：核心变更：新增分块熵函数 vocab_parallel_entropy_with_chunking，提供内存优化的熵计算实现。
- verl/workers/engine/megatron/transformer_impl.py（模块 Megatron 引擎；类别 source；类型 core-logic）：Megatron Engine 主集成点：在 logits_processor 中添加条件分支调用分块熵函数，并更新 import。
- verl/workers/config/engine.py（模块 引擎配置；类别 source；类型 core-logic）：配置模型扩展：在多个 EngineConfig 类中新增分块熵相关配置项。

关键符号：vocab_parallel_entropy_with_chunking

关键源码片段

verl/utils/megatron/tensor_parallel.py

核心变更：新增分块熵函数 vocab_parallel_entropy_with_chunking，提供内存优化的熵计算实现。

```
def vocab_parallel_entropy_with_chunking(
    vocab_parallel_logits: torch.Tensor,
    chunk_size: int = 2048,
) -> torch.Tensor:
    """
    Memory-efficient entropy calculation using chunked processing
    when logits are sharded in tp ranks.

    Args:
        vocab_parallel_logits: (batch_size, seq_len, vocab_size // tp_size) or (total_nnz, vocab_size // tp_size)
        chunk_size: Number of sequence tokens to process at once. Defaults to 2048.

    Returns: (batch_size, seq_len)
    """
    # 准备输出张量，保持与输入相同的 dtype 和设备
    output_shape = list(vocab_parallel_logits.shape[:-1])
    # 使用 torch.empty 而非 torch.zeros 以避免不必要的初始化开销，
    # 并且使用输入 logits 的 dtype，避免 float32 与 bf16/fp16 不匹配
    entropy = torch.empty(output_shape, device=vocab_parallel_logits.device,
                          dtype=vocab_parallel_logits.dtype)

    # 沿序列维度分块，每个 chunk 完整调用 _VocabParallelEntropy
    for i in range(0, vocab_parallel_logits.shape[1], chunk_size):
        logits_chunk = vocab_parallel_logits[:, i : i + chunk_size, :]
        entropy_chunk = _VocabParallelEntropy.apply(logits_chunk)
        entropy[:, i : i + chunk_size] = entropy_chunk

    return entropy
```

verl/workers/engine/megatron/transformer_impl.py

Megatron Engine 主集成点：在 logits_processor 中添加条件分支调用分块熵函数，并更新 import。

```
# ''' 导入新函数（位于文件头部）'''
from verl.utils.megatron.tensor_parallel import (
    vocab_parallel_entropy,
    vocab_parallel_entropy_with_chunking, # 新增导入
    vocab_parallel_log_probs_from_logits,
    vocab_parallel_sum_pi_squared,
)

# ''' 在 logits_processor 中使用（位于 __init__ 的 logits_processor 闭包内）'''
def logits_processor(logits, label, temperature):
    # ... 温度缩放等前置逻辑 ...
    ret = {}
    # sum_pi_squared 必须在 entropy 之前计算，因为其实现是非破坏性的
    if calculate_sum_pi_squared:
        ret["sum_pi_squared"] = vocab_parallel_sum_pi_squared(logits)
    if calculate_entropy:
        # 注意：此处为了向后兼容保留 clone，但分块熵内部使用 empty，不依赖 clone
        logits_bak = logits.clone()
        if self.engine_config.entropy_from_logits_with_chunking:
            # 启用分块熵，通过配置的 chunk_size 控制每块处理的 token 数
            entropy = vocab_parallel_entropy_with_chunking(
                logits,
                chunk_size=self.engine_config.entropy_from_logits_chunk_size,
            )
        else:
            # 默认完整熵计算
            entropy = vocab_parallel_entropy(logits)
        ret["entropy"] = entropy
    else:
        logits_bak = logits
    # ... 后续 log_probs 计算 ...
```

评论区精华

- chunk_size 可配置性：ETOgaosion 建议让 chunk_size 可通过配置编辑，作者 ZLiao097 同意并补充，最终在多个配置类中增加了 entropy_from_logits_chunk_size 字段。
- 分块维度与输入形状：gemini-code-assist[bot] 指出输入若为 2D 会在词表维度分块导致错误；作者解释输入始终为 3D（来自 Megatron 的 ColumnParallelLinear），因此分块维度正确，且已在回复中澄清。
- dtype 一致性：duesdues 指出 torch.zeros 默认 float32 与 logits 的 bf16/fp16 不匹配，作者在后续提交中修复为使用输入 dtype。
- chunk_size 参数传递：zyang6 指出在未启用分块时仍传递 chunk_size 可能导致 TypeError，建议使用 functools.partial 或在调用点加判断，作者回复已解决。

- chunk_size 可配置性 (design): 作者同意并在后续提交中在多个配置类中增加了 entropy_from_logits_chunk_size 字段。
- 分块维度与输入形状正确性 (correctness): 作者解释输入始终是 3D（来自 Megatron 的 ColumnParallelLinear），因此分块维度正确。
- dtype 一致性 (correctness): 作者在后续提交中修复为使用输入 logits 的 dtype。
- chunk_size 参数传递冲突 (design): 作者回复已解决（通过添加 if 分支避免传递 chunk_size）。

风险与影响

- 风险：
  - 兼容性风险：新增条件分支默认关闭，不影响现有流程，但仍需确保无意外破坏。
  - 正确性风险：分块计算熵在数学上等价于整体计算，但初始化时默认 float32 的问题已被修复，当前版本已使用输入 dtype。
  - 性能风险：分块循环引入轻微开销，但内存收益显著；默认 chunk_size=2048 需实际调优。
  - 配置重复风险：FSDPEngineConfig 等类中已存在 entropy_from_logits_with_chunking 字段，本次又在同一类中重复添加（patch 显示新增该行），可能导致重复定义错误，需确认。
  - 缺少测试覆盖：本次改动未包含直接对应的测试文件变更，存在回归风险。
- 影响：
  - 用户影响：为 Megatron 用户提供显存优化开关，只需在配置中设置 entropy_from_logits_with_chunking: true 并可选调整 chunk_size；其他后端用户无影响。
  - 系统影响：主要降低长序列训练时的显存峰值，有助于在有限 GPU 显存下处理更长上下文。
  - 团队影响：新增函数和配置需在文档中说明，且缺少测试覆盖，后续需补充。
  - 风险标记：缺少测试覆盖，配置重复定义风险，兼容性风险低

关联脉络

- 暂无明显关联 PR