PR #26658 完整报告

sgl-project/sglang

test: strengthen CG-replay coverage with prod-fill padding, metadata invariants, and pad-ratio sweep

合并时间：2026-05-29 13:43

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26658

执行摘要

- 一句话：强化 CG-replay 测试：生产填充、元数据不变式、多比例扫描
- 推荐动作：值得精读。本 PR 展示了如何通过分析测试 / 生产环境差异来设计有针对性的测试覆盖。assert_cg_metadata_well_formed 的设计原则（best-effort、静默跳过、单语句检查）和 pad_style 抽象值得在其他测试套件中复用。

功能与动机

Codex review of #26651 发现了一个真实 bug（CG replay 时 kv_lens 为负），而现有测试因填充模式不匹配未能捕获。根本原因是测试用的填充模式（padded 行的 seq_lens 设为 capture_prefix_len + N）与生产环境（padded 行的 seq_lens 设为 seq_len_fill_value、extend_seq_lens 设为 num_tokens_per_bs，导致减出来为负）不同。本 PR 旨在缩小测试与生产环境之间的差距，并增加两层防御。

实现拆解

1. 新增 metadata_invariants.py（python/sglang/test/kits/attention_unittest/runner_modes/metadata_invariants.py）：定义 assert_cg_metadata_well_formed 函数，在 CG replay 元数据初始化后执行，检查 forward_metadata 上的 indptr 数组是否单调非递减且首元素为 0、per-request 长度是否非负。支持 FA 和 Triton 后端，缺失字段时静默跳过。
2. 修改 speculative_cuda_graph_runner.py：为 SpeculativeCudaGraphAdapter 类新增 pad_style 字段（Literal["small_real", "prod_fill"]）和 pad_num_tokens_per_bs 字段；实现 _apply_prod_fill_padding 函数，按生产版本的方式覆写 padded 行的 seq_lens、extend_seq_lens、req_pool_indices、out_cache_loc、positions、input_ids 等，使 padded 行的 seq_lens - extend_seq_lens 为负，考验 backends 的鲁棒性。
3. 修改各测试文件（test_fa3.py、test_fa4.py、test_triton.py、test_mla_triton.py）中的 DRAFT_EXTEND_V2 测试方法：从单个 subTest 扩展为 pad_style ∈ {small_real, prod_fill} × capture_bs ∈ {1×, 2×, 4×} 的 6 个 subTest，覆盖 0%、50%、75% 填充比例。
4. 配套调整：在 cuda_graph_decode_runner.py 中导入并调用 assert_cg_metadata_well_formed；在 speculative_draft_extend_runner.py 中添加 pad_style 参数透传。

关键文件：

- python/sglang/test/kits/attention_unittest/runner_modes/metadata_invariants.py（模块 测试工具；类别 test；类型 test-coverage；符号 _slice_bs_plus_one, _slice_bs, assert_cg_metadata_well_formed）：新增文件，定义后端无关的 CG replay 元数据不变式检查函数 assert_cg_metadata_well_formed，是本次测试强化的核心工具。
- python/sglang/test/kits/attention_unittest/runner_modes/speculative_cuda_graph_runner.py（模块 测试 runner；类别 test；类型 test-coverage；符号 _apply_prod_fill_padding, PadStyle）：修改核心 runner，新增生产风格填充函数和 PadStyle 类型，是测试生产模拟的关键入口。
- test/registered/attention/unittests/dense/test_fa3.py（模块 FA3 测试；类别 test；类型 test-coverage）：DRAFT_EXTEND_V2 测试用例被扩展为 pad_style×capture_bs 的 6 个 subTest，是全套测试中覆盖比率扫描的主要场景之一。
- test/registered/attention/unittests/dense/test_fa4.py（模块 FA4 测试；类别 test；类型 test-coverage）：同上，FA4 后端同等扩展。
- test/registered/attention/unittests/dense/test_triton.py（模块 Triton 测试；类别 test；类型 test-coverage）：Triton dense 后端增加 pad_style 和 capture_bs 参数。
- test/registered/attention/unittests/mla/test_triton.py（模块 MLA 测试；类别 test；类型 test-coverage）：MLA Triton 后端同样增加 pad_style 和 capture_bs 参数。
- python/sglang/test/kits/attention_unittest/runner_modes/speculative_draft_extend_runner.py（模块 扩展 runner；类别 test；类型 test-coverage）：透传 pad_style 参数到下游函数，使 prod_fill 模式生效。
- python/sglang/test/kits/attention_unittest/runner_modes/cuda_graph_decode_runner.py（模块 解码 runner；类别 test；类型 test-coverage）：在 replay 元数据初始化后调用 assert_cg_metadata_well_formed，确保所有 CG 路径都能受益。

关键符号：assert_cg_metadata_well_formed, _apply_prod_fill_padding, _slice_bs_plus_one, _slice_bs

关键源码片段

python/sglang/test/kits/attention_unittest/runner_modes/metadata_invariants.py

新增文件，定义后端无关的 CG replay 元数据不变式检查函数 assert_cg_metadata_well_formed，是本次测试强化的核心工具。

```
"""Backend-agnostic assertions on `forward_metadata` after CG replay init.

Catches corruption that the output-equality assertion misses — e.g., negative
per-request lengths or non-monotonic indptr that happen to leave real-row
output correct while corrupting padded-row scratch state.

Usage from a CG runner kit:

    from .metadata_invariants import assert_cg_metadata_well_formed
```

```
    _init_cuda_graph_replay_metadata(backend, capture_batch_size, replay_batch)
    assert_cg_metadata_well_formed(backend, bs=capture_batch_size)
"""

from __future__ import annotations
from typing import Any
import torch

# CSR indptr 字段名列表，期望非递减且首元素为 0
_INDPTR_FIELDS = (
    "kv_indptr", "qo_indptr", "mask_indptr", "window_kv_indptr",
    "cu_seqlens_q", "cu_seqlens_k", "encoder_cu_seqlens_k",
)

# 每个请求的长度字段列表，期望非负
_PER_REQ_LEN_FIELDS = (
    "cache_seqlens_int32", "encoder_lens_int32", "local_seqused_k",
    "max_seq_len_k", # 可能是标量，防御性检查
)

def _slice_bs_plus_one(t: torch.Tensor, bs: int) -> torch.Tensor:
    # indptr 长度为 bs+1，有些后端预分配了 max_bs+1，取前 bs+1 个元素
    return t[: bs + 1] if t.numel() >= bs + 1 else t

def _slice_bs(t: torch.Tensor, bs: int) -> torch.Tensor:
    return t[:bs] if t.numel() >= bs else t

def assert_cg_metadata_well_formed(backend: Any, bs: int) -> None:
    """检查 backend.forward_metadata 是否明显损坏。
    Best-effort: 仅当字段存在且为 tensor 时才检查。
    """
    meta = getattr(backend, "forward_metadata", None)
    if meta is None:
        return

    errors: list[str] = []

    # 检查所有 indptr 字段：非递减且首元素为 0
    for field in _INDPTR_FIELDS:
        t = getattr(meta, field, None)
        if not isinstance(t, torch.Tensor):
            continue
        sliced = _slice_bs_plus_one(t, bs)
        if sliced.numel() < 2:
            continue
        diff = sliced[1:].to(torch.int64) - sliced[:-1].to(torch.int64)
        # 允许零长度请求 (diff==0)，拒绝负 diff
        if (diff < 0).any().item():
            min_diff = diff.min().item()
```

```
                errors.append(f"{field} 在 bs={bs} 处非单调递减 (相邻最小差={min_diff}); 切片={sliced[:
                min(bs+1,16)].tolist()}")
            if sliced[0].item() != 0:
                errors.append(f"{field}[0] != 0 (为 {sliced[0].item()}) 在 bs={bs}")

    # 检查所有 per-request 长度字段：非负
    for field in _PER_REQ_LEN_FIELDS:
        t = getattr(meta, field, None)
        if not isinstance(t, torch.Tensor):
            continue
        sliced = _slice_bs(t, bs)
        if sliced.numel() == 0:
            continue
        if (sliced < 0).any().item():
            min_v = sliced.min().item()
            errors.append(f"{field} 在 bs={bs} 处出现负值 (最小值={min_v}); 切片={sliced[:min(bs,16)].
            tolist()}")

    if errors:
        raise AssertionError(
            "CG forward_metadata 不变式在 replay init 后被违反:
- "
            + "
- ".join(errors)
        )
```

python/sglang/test/kits/attention_unittest/runner_modes/speculative_cuda_graph_runner.py

修改核心 runner，新增生产风格填充函数和 PadStyle 类型，是测试生产模拟的关键入口。

```
def _apply_prod_fill_padding(
    batch,
    *,
    real_bs: int,
    capture_bs: int,
    seq_len_fill_value: int,
    num_tokens_per_bs: int,
) -> None:
    """按照生产 CG runner 的方式覆写 batch 中填充行的元数据。

    生产环境将填充行设置为：seq_lens=fill, extend_seq_lens=N,
    req_pool_indices=0, out_cache_loc=0, positions=0,
    input_ids=0。这样 seq_lens - extend_seq_lens 会变为负值，
    后端必须对此进行防御（如 clamp）。
    """
    if real_bs >= capture_bs:
        return
    pad_lo, pad_hi = real_bs, capture_bs
```

```
# 覆写 per-request 长度张量
batch.seq_lens[pad_lo:pad_hi] = seq_len_fill_value
if batch.seq_lens_cpu is not None:
    batch.seq_lens_cpu[pad_lo:pad_hi] = seq_len_fill_value
    batch.seq_lens_sum = int(batch.seq_lens_cpu.sum())

if getattr(batch, "extend_seq_lens", None) is not None:
    batch.extend_seq_lens[pad_lo:pad_hi] = num_tokens_per_bs
if getattr(batch, "extend_seq_lens_cpu", None) is not None:
    ext = list(batch.extend_seq_lens_cpu)
    for i in range(pad_lo, min(pad_hi, len(ext))):
        ext[i] = num_tokens_per_bs
    batch.extend_seq_lens_cpu = ext

# 覆写 per-request slot 张量
batch.req_pool_indices[pad_lo:pad_hi] = 0

# 覆写 per-token 张量：填充行占据 [real_bs*N, capture_bs*N) 切片
tok_lo = pad_lo * num_tokens_per_bs
tok_hi = pad_hi * num_tokens_per_bs
for field in ("out_cache_loc", "positions", "input_ids"):
    t = getattr(batch, field, None)
    if t is not None and t.numel() >= tok_hi:
        t[tok_lo:tok_hi] = 0

# 同步 spec_info 中的 extend_seq_lens_tensor
spec_info = getattr(batch, "spec_info", None)
if spec_info is not None:
    eslt = getattr(spec_info, "extend_seq_lens_tensor", None)
    if isinstance(eslt, torch.Tensor) and eslt.numel() >= pad_hi:
        eslt[pad_lo:pad_hi] = num_tokens_per_bs
    eslc = getattr(spec_info, "extend_seq_lens_cpu", None)
    if isinstance(eslc, (list, tuple)) and len(eslc) >= pad_hi:
        ext2 = list(eslc)
        for i in range(pad_lo, pad_hi):
            ext2[i] = num_tokens_per_bs
        spec_info.extend_seq_lens_cpu = ext2
```

评论区精华

PR 无 review 评论。PR 作者在 body 中详细解释了设计决策：选择生产风格填充而非简单模拟，是因为真实生产场景中 padded 行元数据不一致（seq_lens 和 extend_seq_lens 组合会导致负值），必须被后端正确处理。metadata_invariants 的检查是 best-effort 的，不会导致错误阻断，但对已暴露的字段执行强检查。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。本 PR 仅修改测试代码，不影响生产路径。但需注意：
  - 新增的 assert_cg_metadata_well_formed 可能在 debug 模式下引入额外开销，但每个 CG replay 只调用一次，影响可忽略。
  - pad_style="prod_fill" 仅在 DRAFT_EXTEND_V2 路径使用，其他 CG 路径（TARGET_VERIFY、DRAFT_EXTEND_V1）未覆盖，未来需扩展。
  - FA3/FA4 的 DRAFT_EXTEND_V2 测试仍被 skipTest 跳过（已知问题），测试强化不影响现有 skip 逻辑。
  - 影响：影响范围：仅限 Attention Unittest 套件（4-gpu-b200 CI）。新增约 10 个 subTest，总 subTest 数从 525 增至 535。对 CUDA Graph replay 路径的测试置信度显著提升，能捕获负 kv_lens 等之前遗漏的 bug。团队可参考 metadata_invariants.py 的设计模式为其他模块添加类似的不变式检查。
  - 风险标记：仅覆盖 DRAFT_EXTEND_V2，新增断言可能假阳性但可忽略

关联脉络

- PR #26655 Fix TRTLLM MHA draft decode cache seqlens replay: 同为 CG replay 相关的 bugfix，本 PR 的测试增强可防止同类问题
- PR #26628 Revert "Fix FA DRAFT_EXTEND_V2 cache extent": 涉及 DRAFT_EXTEND_V2 cache extent 修复的回滚，本 PR 增强了该路径的测试覆盖