PR #26517 完整报告

sgl-project/sglang

Add attention-backend unit-test suite under test/registered/attention/unittest

合并时间：2026-05-29 08:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26517

执行摘要

- 一句话：为注意力后端添加模块级单元测试套件
- 推荐动作：建议所有关注注意力后端的开发者仔细阅读本 PR 的测试架构，特别是参考实现的设计和 SWA *decode* 规则分类。后续新增后端时，务必在 *dense_attention.py* 中注册 SWA *decode* 规则，并按照已有模式添加测试文件。KNOWN_FAILURES.md 也是必读文档，用于理解当前后端限制。

功能与动机

现有 *test/registered/attention/* 下的测试为服务器级端到端测试，后端回归时失败晚且难以隔离。本 PR 添加模块级正确性套件，在孤立环境中测试每个支持组合，失败时直接指向具体的后端方法和输入形状。

实现拆解

1. 在 python/sglang/test/kits/ 下建立共享测试工具库，包括 mock_server_args.py、*per-method fixture* 和 *runner modes*。
2. 在 *test/registered/attention/unittest/* 下按 *attention method* 组织测试文件，每个测试文件对应一个后端，通过构建独立的 PyTorch 参考模块进行比较。
3. 实现覆盖维度包括：*attention methods*（密集、SWA、MLA、DSA、DSV4、GDN、KDA、Lightning、*Mamba2* 等）、*backends*（FA3、FA4、FlashInfer、Triton、TorchNative 等）、*forward modes*（DECODE、EXTEND、MIXED、TARGET_VERIFY、DRAFT_EXTEND 等）、*runner modes*（*eager*、CUDA-graph、split-op、EAGLE 等）、*speculative kinds*（EAGLE *chain/tree*、FrozenKV MTP、DFlash、NGRAM）、*input shapes*（页边界、前缀精确页、不规则 batch 等）、*cache layouts*（*contiguous*、*shuffled_pages*、*interleaved_pages*、*non_monotonic_extend*）。
4. 添加 KNOWN_FAILURES.md 文档记录每个已知后端限制及其需要采取的行动，以及 *per-method* README.md。
5. 在 CI 中通过 *base-b-test-4-gpu-b200* 和 *base-b-test-1-gpu-large* 两个 *suite* 注册测试，无需更改 workflow 文件。

关键文件：

- python/sglang/test/kits/attention_unittest/attention_methods/*dense_attention.py*（模块 密集注意力；类别 test；类型 *test-coverage*；符号 *_swa_decode_uses_min_seq_len_rul*

e, DenseAttentionCase, batch_size, input_lens）：提供基础注意力案例 dataclass 定义和 SWA decode 规则分类，是所有密集注意力测试的核心基础。

- python/sglang/test/kits/attention_unittest/attention_methods/dsa_attention.py（模块 DSA 注意力；类别 test；类型 test-coverage；符号 DSAAttentionCase, make_dsa_dense_fallback_cases, make_dsa_sparse_cases, TinyDSAModelConfig）：定义 DSA 注意力密集回退和稀疏测试案例，覆盖页边界条件，是 DSA 后端测试的基础。
- python/sglang/test/kits/attention_unittest/attention_methods/dsv4_attention.py（模块 DSV4 注意力；类别 test；类型 test-coverage；符号 DSV4AttentionCase, batch_size, input_lens, seq_lens）：提供 DSV4 注意力 SWA-only 切片的测试案例，包含 FP8 量化与容差处理，是 DSV4 后端测试的核心。
- python/sglang/test/kits/attention_unittest/runner_modes/speculative_draft_extend_runner.py（模块 推测解码；类别 test；类型 test-coverage；符号 _make_dense_spec_case_with_lens, _make_eagle_draft_extend_input, _make_frozen_kv_mtp_draft_extend_input, _make_draft_extend_input）：实现推测解码 draft-extend 运行器的测试辅助，覆盖 EAGLE、FrozenKV MTP、EAGLE v2 等模式，是推测解码测试的关键。

关键符号：_swa_decode_uses_min_seq_len_rule, make_dense_input_config_cases, make_dense_attention_config_cases, make_dsa_dense_fallback_cases, make_dsa_sparse_cases, make_dsv4_cases, _make_eagle_draft_extend_input, _make_frozen_kv_mtp_draft_extend_input, _make_draft_extend_input, _make_eagle_draft_extend_v2_input, _prepare_draft_extend_batch, _prepare_eagle_draft_extend_batch

关键源码片段

python/sglang/test/kits/attention_unittest/attention_methods/dense_attention.py

提供基础注意力案例 dataclass 定义和 SWA decode 规则分类，是所有密集注意力测试的核心基础。

```
# 分类 SWA decode 使用的元数据规则，因生产后端不同而异。
# - min_seq_len_window: window_kv_lens = min(seq_lens, window)（不含当前 token）
# - extend_window: 允许 [query_pos - window, query_pos] 的 key（含当前 token）
_SWA_DECODE_MIN_SEQ_LEN_WINDOW: frozenset[str] = frozenset({"triton"})
_SWA_DECODE_EXTEND_WINDOW: frozenset[str] = frozenset(
    {"torch_native", "fa3", "fa4", "flex_attention", "trtllm_mha", "flashinfer"}
)

def _swa_decode_uses_min_seq_len_rule(case: "DenseAttentionCase") -> bool:
    if case.backend in _SWA_DECODE_MIN_SEQ_LEN_WINDOW:
        return True
    if case.backend in _SWA_DECODE_EXTEND_WINDOW:
        return False
    raise ValueError(
        f"未知的 SWA decode 规则后端 {case.backend!r}。请根据其 "
```

```
        f"`init_forward_metadata_decode` 生成的元数据，将其添加到 "
        f"`_SWA_DECODE_MIN_SEQ_LEN_WINDOW` 或 `_SWA_DECODE_EXTEND_WINDOW`。"
    )

@dataclass(frozen=True)
class DenseAttentionCase:
    name: str
    backend: str
    forward_mode: ForwardMode
    num_heads: int
    num_kv_heads: int
    page_size: int
    prefix_lens: tuple[int, ...]
    extend_lens: tuple[int, ...] = ()
    sliding_window_size: int | None = None

    @property
    def batch_size(self) -> int:
        return len(self.prefix_lens)

    @property
    def input_lens(self) -> tuple[int, ...]:
        if self.forward_mode.is_decode():
            return (1,) * self.batch_size
        return self.extend_lens

    @property
    def seq_lens(self) -> tuple[int, ...]:
        return tuple(p + q for p, q in zip(self.prefix_lens, self.input_lens))

    @property
    def num_input_tokens(self) -> int:
        return sum(self.input_lens)
```

python/sglang/test/kits/attention_unittest/attention_methods/dsa_attention.py

定义 DSA 注意力密集回退和稀疏测试案例，覆盖页边界条件，是 DSA 后端测试的基础。

```
@dataclass(frozen=True)
class DSAAttentionCase(DenseAttentionCase):
    pass

DSA_PAGE_SIZE = 64
DSA_INDEX_TOPK = 8
DSA_SPARSE_ATOL = 1.6e-1
DSA_SPARSE_RTOL = 1.6e-1

def make_dsa_dense_fallback_cases(backend: str) -> tuple[DSAAttentionCase, ...]:
    common = dict(
```

```
        backend=backend,
        forward_mode=ForwardMode.EXTEND,
        num_heads=4,
        num_kv_heads=4,
        page_size=DSA_PAGE_SIZE,
    )
    return (
        DSAAttentionCase(
            name="dsa_mha_one_shot_no_prefix_ragged",
            prefix_lens=(0, 0, 0),
            extend_lens=(3, 8, 17),
            **common,
        ),
        DSAAttentionCase(
            name="dsa_mha_one_shot_no_prefix_exact_page",
            prefix_lens=(0,),
            extend_lens=(DSA_PAGE_SIZE,),
            **common,
        ),
        # 零前缀 extend，序列长度刚好比页大小少 1，与精确页和跨页案例组合覆盖 < 页、= 页、> 页
        三种情况
        DSAAttentionCase(
            name="dsa_mha_one_shot_no_prefix_seq_below_page",
            prefix_lens=(0,),
            extend_lens=(DSA_PAGE_SIZE - 1,),
            **common,
        ),
        # 混合 batch：低于、等于、高于页边界，测试密集回退路径的 K 写入不跨请求页表
        DSAAttentionCase(
            name="dsa_mha_one_shot_ragged_below_at_above_page",
            prefix_lens=(0, 0, 0),
            extend_lens=(DSA_PAGE_SIZE - 1, DSA_PAGE_SIZE, DSA_PAGE_SIZE + 1),
            **common,
        ),
    )
```

评论区精华

讨论 1：unittest 目录命名冲突

michaelzhang-ai 指出 test/registered/attention/unittest/__init__.py 使该目录成为普通包，当注册测试运行器执行每个测试时，该目录位于 sys.path[0]，导致标准库 unittest 被阴影，引发 import unittest.mock 等错误。作者确认已在 PR #26654 中解决。

讨论 2：CI 注册 review

chatgpt-codex-connector[bot] 建议添加 register_cuda_ci 注册以避免 CI 发现 abort。作者在 PR body 中说明已通过 base-b-test-4-gpu-b200 和 base-b-test-1-gpu-large 两个 suite 自动注册，无需额外改动。

- unittest 目录命名导致标准库阴影 (correctness): 作者在 PR #26654 中通过调整解决了此问题。
- 测试文件 CI 注册缺失提醒 (testing): 作者在 PR body 中说明已通过 *base-b-test-4-gpu-b200* 和 *base-b-test-1-gpu-large* 两个 suite 自动注册，无需额外改动。

风险与影响

- 风险：
  1. 命名冲突风险：已将目录命名为 unittest，与标准库冲突，已在 #26654 修复，但类似的命名问题仍需警惕。
  2. CI 时间增加：27 个测试文件预计约 595 秒，可能延长 CI 流水线总时长。
  3. 硬件依赖：部分测试需要 Blackwell GPU（SM>=100），在其他硬件上自动跳过，但可能降低非 Blackwell 环境的覆盖率。
  4. 参考实现漂移：随着后端变化，PyTorch 参考可能与实际后端行为偏离，需要维护。 - 影响：对开发者：当引入新的注意力后端或修改现有后端时，可以立即运行对应的单元测试来验证正确性，大幅缩短回归定位时间。对系统：减少生产环境出现注意力计算错误的风险。对团队：建立了统一的测试框架，新增后端时只需按照 fixture 模式添加测试文件和案例即可。
- 风险标记：命名冲突，CI 时间增加，硬件依赖

关联脉络

- PR #26512 Fix FA DRAFT_EXTEND_V2 *cache extent*: 本 PR 测试覆盖的基础 bug 修复之一
- PR #26513 Fix FlashInfer SWA EXTEND-with-prefix *correctness*: 本 PR 测试覆盖的基础 bug 修复之一
- PR #26514 Expose Flex *attention causal/decode masks as static methods*: 本 PR 测试覆盖的基础 bug 修复之一
- PR #26515 Allow Optional *key/value* in unified_attention_with_output (MLA *absorb* fix): 本 PR 测试覆盖的基础 bug 修复之一
- PR #26516 Add sliding-window mask support to TorchNative*AttnBackend*: 本 PR 测试覆盖的基础 bug 修复之一
- PR #26654 Fix unittest *directory naming conflict*: 修复本 PR 引入的 unittest 目录命名冲突