PR #33641 完整报告

sgl-project/sglang

[CI] Merge tokenizer worker tests and drop redundant triton attention e2e

合并时间：2026-08-06 02:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33641

执行摘要

- 一句话：合并 tokenizer 测试、删冗余 e2e，H100 CI 减时约 13 分钟
- 推荐动作：值得精读 PR body 的覆盖等价性论证与 test_int8_linear_methods.py 的实现模式：真实层 + weight_loader_v2 + 手写 dequant oracle，这种 "e2e 冗余 → 层 UT 下沉 " 的方法对后续量化 / 后端测试设计有借鉴价值。建议关注合并后 base-b/base-c 套件的稳定性，并考虑为 triton/flashinfer 的端到端精度阈值补充轻量替代或文档说明。

功能与动机

PR body 明确指出目标：'Trims on the per-commit H100 suites (base-b 1-gpu ~ -548s, base-c 4-gpu -220s), replacing redundant e2e matrices with existing or new layer-level coverage'。作者逐条论证了各被删用例的覆盖等价性：detokenizer 的 ttft 测试与 tokenizer 仅差一个 flag；triton attention 数值已由 attention/unittests 覆盖；gpt-oss bf16 与 mxfp4 版本字节级相同；MLA per-backend 数值已由 attention/unittests/mla 覆盖。此外删除了本就 skipIf(is_in_ci()) 的 block-int8 死代码，把量化轴回归信号下沉到新的层 UT。

实现拆解

1. 合并分词器测试：test/registered/tokenizer/test_multi_tokenizer.py 的 setUpClass 追加 --detokenizer-worker-num 4，与原有 --tokenizer-worker-num 8 并跑；删除 test_multi_detokenizer.py。两个 flag 正交，单次 server 启动即可同时覆盖两条 worker 池路径，节省约 211s，MMLU 阈值与 ttft/ITL 断言保持不变。
2. 删除 triton attention e2e：test_triton_attention_backend.py 整体删除；mmlu 端到端精度断言由 attention/unittests/{dense,swa,gdn}/test_triton.py 的数值测试替代，offline throughput > 153 断言由 perf/test_bench_serving_1gpu_part1.py::test_offline_throughput_with_triton_attention_backend 承担，节省约 177s。
3. 删除 gpt-oss bf16 双胞胎：test_gpt_oss_4gpu_bf16.py 与 mxfp4 版本仅 quantization="bf16" 不同；保留 mxfp4 的 h100/b200 e2e 及 test_mxfp4_sm90_cutlass.py 层 UT，bf16 upcast 路径改为人工 triage 时使用，节省约 220s。
4. MLA e2e 收敛 + INT8 单测补位：删除 test_flashmla.py 与 test_mla_flashinfer.py（per-backend MLA 数值已由 attention/unittests/mla/test_{flashmla,flashinfer}.py 覆盖）；test_mla_int8_deepseek_v3.py 保留为 channel-int8 + MTP 默认后端 smoke 并移除 skipIf(is_in_ci()) 的 TestDeepseekV3MTPBlockInt8 死代码；新增

test_int8_linear_methods.py，通过真实 ColumnParallelLinear + weight_loader_v2 加载手写量化权重，对比反量化参考矩阵，覆盖 W8A8Int8LinearMethod 与 BlockInt8LinearMethod 两种方法，约 +60s。

5. CI 配套：新 UT 注册 stage base-b、1-gpu-large、est_time 60s；pr-test-extra 曾失败一次（Run #30990840598），作者 rerun 三个关键测试后通过，确认新矩阵在 H100 上稳定。

关键文件：

- test/registered/unit/layers/quantization/test_int8_linear_methods.py（模块 量化单测；类别 test；类型 test-coverage；符号 _quantize_int8_channel, _quantize_int8_block, _Int8LinearCheck, _check）：本次唯一新写入的测试资产，用真实 ColumnParallelLinear + weight_loader_v2 覆盖 channel W8A8 与 blockwise int8 两种量化方法，并替代了原先在 CI 中被 skip 的 block-int8 死代码。
- test/registered/mla/test_flashmla.py（模块 注意力测试；类别 test；类型 deletion；符号 TestFlashMLAMTP, setUpClass, tearDownClass, test_gsm8k）：删除的 FlashMLA + EAGLE MTP e2e，作者认为其 per-backend 数值已由 attention/unittests/mla/test_flashmla.py 覆盖，节省约 160s。
- test/registered/mla/test_mla_flashinfer.py（模块 注意力测试；类别 test；类型 deletion；符号 TestFlashinferMLAMTP, test_gsm8k）：删除的 FlashInfer MLA + EAGLE MTP e2e，与 flashmla 版本同为 per-backend 数值覆盖，由 attention/unittests/mla/test_flashinfer.py 承接。
- test/registered/tokenizer/test_multi_detokenizer.py（模块 分词器测试；类别 test；类型 deletion；符号 TestMultiDetokenizer, setUpClass, tearDownClass, test_multi_detokenizer_ttft）：删除；其 ttft 测试与 test_multi_tokenizer.py 仅差 --detokenizer-worker-num 参数，合并后由同一 server 覆盖，节省约 211s。
- test/registered/attention/test_triton_attention_backend.py（模块 注意力测试；类别 test；类型 deletion；符号 TestTritonAttnBackend, test_latency, test_mmlu）：删除的 triton attention e2e；数值由 attention/unittests 承接，吞吐断言由 perf 套件承接，节省约 177s。
- test/registered/mla/test_mla_int8_deepseek_v3.py（模块 量化冒烟；类别 test；类型 test-coverage；符号 TestDeepseekV3MTPChannelInt8, TestDeepseekV3MTPBlockInt8, test_gsm8k）：修改；删除 skipIf(is_in_ci()) 的 TestDeepseekV3MTPBlockInt8 死代码，保留 channel-int8 + MTP 默认后端 smoke，作为 MLA e2e 收敛后的唯一服务级回归。
- test/registered/models_e2e/test_gpt_oss_4gpu_bf16.py（模块 多卡测试；类别 test；类型 deletion；符号 TestGptOss4GpuBf16, test_bf16_120b）：删除的 gpt-oss 120b bf16 4-GPU e2e；与 mxfp4 版本字节级相同，保留原生格式覆盖，节省约 220s。
- test/registered/tokenizer/test_multi_tokenizer.py（模块 分词器测试；类别 test；类型 test-coverage；符号 TestMultiTokenizer, setUpClass）：修改；合并多 detokenizer 覆盖，server 启动同时带两个 worker flag。

关键符号：_quantize_int8_channel, _quantize_int8_block, _Int8LinearCheck._check, TestW8A8Int8Linear.test_channel, TestBlockInt8Linear.test_block,

TestMultiTokenizer.setUpClass, TestDeepseekV3MTPChannelInt8.test_gsm8k,TestGptOss4GpuBf16.test_bf16_120b, TestFlashMLAMTP.test_gsm8k,TestFlashinferMLAMTP.test_gsm8k

关键源码片段

test/registered/unit/layers/quantization/test_int8_linear_methods.py

本次唯一新写入的测试资产，用真实 ColumnParallelLinear + weight_loader_v2 覆盖 channel W8A8 与 blockwise int8 两种量化方法，并替代了原先在 CI 中被 skip 的 block-int8 死代码。

```
# 量化辅助函数：把随机浮点权重转为 checkpoint 格式的 int8 + scale，
# 同时返回反量化参考矩阵，作为后续数值校验的 oracle。
def _quantize_int8_channel(w: torch.Tensor):
    # per-output-channel 对称量化，scale 形状为 [N, 1]
    amax = w.float().abs().amax(dim=1, keepdim=True).clamp(min=1e-12)
    scale = amax / INT8_MAX
    w_int8 = torch.round(w.float() / scale).clamp(-INT8_MAX, INT8_MAX).to(torch.int8)
    w_dequant = w_int8.float() * scale
    return w_int8, scale, w_dequant


def _quantize_int8_block(w: torch.Tensor, block: int = 128):
    # per (128, 128) tile 对称量化，scale 形状为 [N / block, K / block]
    n, k = w.shape
    tiles = w.float().reshape(n // block, block, k // block, block)
    amax = tiles.abs().amax(dim=(1, 3)).clamp(min=1e-12)
    scale = amax / INT8_MAX
    w_int8 = (
        torch.round(tiles / scale[:, None, :, None])
        .clamp(-INT8_MAX, INT8_MAX)
        .to(torch.int8)
    )
    w_dequant = (w_int8.float() * scale[:, None, :, None]).reshape(n, k)
    return w_int8.reshape(n, k), scale, w_dequant


class _Int8LinearCheck(CustomTestCase):
    @classmethod
    def setUpClass(cls):
        # 初始化单进程分布式环境，供真实层加载使用
        init_single_process_dist()

    def _check(self, shapes, build_layer):
        torch.manual_seed(7)
        for m, n, k in shapes:
            with self.subTest(shape=(m, n, k)):
                # 走真实 ColumnParallelLinear + weight_loader_v2 路径，
```

```
            # 而不是直接调用 kernel，以覆盖权重加载与量化方法装配。
            layer, w_dequant = build_layer(n, k)
            layer.quant_method.process_weights_after_loading(layer)
            x = torch.randn((m, k), device="cuda", dtype=torch.bfloat16) / 10
            out, _ = layer(x)
            ref = x.float() @ w_dequant.T
            # atol 吸收动态 per-token activation 量化引入的误差
            assert_output_close(self, out, ref, rtol=5e-2, atol=1e-1)


@unittest.skipIf(
    get_device_sm() >= 100, "sgl-kernel int8_scaled_mm has no SM100+ kernel"
)
class TestW8A8Int8Linear(_Int8LinearCheck):
    @staticmethod
    def _build_layer(n: int, k: int):
        layer = make_tp1_column_parallel_linear(W8A8Int8Config({}), n, k)
        w = torch.randn((n, k), device="cuda", dtype=torch.bfloat16) / 10
        w_int8, scale, w_dequant = _quantize_int8_channel(w)
        load_linear_weights(layer, weight=w_int8, weight_scale=scale)
        return layer, w_dequant

    def test_channel(self):
        self._check(CHANNEL_SHAPES, self._build_layer)
```

评论区精华

本 PR 没有产生实质 review 评论，核心论证都在 PR body 中：

> triton attention numerics are covered by attention/unittests/{dense,swa,gdn}/test_triton.py on the same runner... the offline-throughput assertion is covered by perf/test_bench_serving_1gpu_part1.py::test_offline_throughput_with_triton_attention_backend
>
> test_gpt_oss_4gpu_bf16.py ... byte-identical to test_gpt_oss_4gpu_mxfp4.py except quantization="bf16"
>
> Drop TestDeepseekV3MTPBlockInt8 from test_mla_int8_deepseek_v3.py: it was already skipIf(is_in_ci()) dead code

唯一的交互是作者发起的 /rerun-test，bot 返回："🤔 1-gpu-h100 (3 tests): 🤔"。未解决的疑虑：triton *e2e* 中 mmlu > 0.65 的端到端精度断言没有在 PR 内展开讨论是否有直接替代。

- 覆盖等价性论证：*e2e* 冗余如何下沉到 layer 级测试 (design): 作者认为等价覆盖成立，合并后无需保留这些 *e2e*；未收到 reviewer 反对意见。
- 合并后新矩阵的 rerun 验证 (testing): bot 返回 1-gpu-h100 三个测试全部通过，新矩阵在 H100 上稳定。

风险与影响

- 风险：覆盖缺口：triton e2e 中 mmlu > 0.65 的端到端精度断言没有明确替代（perf 套件只验吞吐）；flashinfer + EAGLE 组合的 avg_spec_accept_length > 2.5 断言消失，组合级回归可能延迟暴露；gpt-oss bf16 upcast 路径失去 CI 覆盖，只能手动 triage。新 UT 局限：test_int8_linear_methods.py 在 SM100+ 上整体 skip（sgl-kernel int8_scaled_mm 无 SM100+ kernel），Blackwell 后续架构缺少 INT8 GEMM 数值回归；断言较宽松（rtol=5e-2、atol=1e-1），对 kernel 系统性偏差的敏感度有限。本 PR 无运行时源码改动，因此不引入生产代码回归风险。
- 影响：用户无感知，不涉及任何 sglang 运行时 /API 变更。团队侧：base-b 1-gpu 套件减时约 548s（约 9 分钟）、base-c 4-gpu 减时约 220s（约 3.7 分钟），tokenizer 测试的 server 启动次数从 2 次降为 1 次。新增 INT8 层 UT（约 60s）为 channel/block int8 提供快速回归信号，替代原先在 CI 中被 skip 的死代码。影响范围主要限于 test/ 目录下的 CI 注册与测试资产。
- 风险标记：e2e 精度断言下沉后无直接替代，flashinfer+EAGLE 组合覆盖移除，SM100+ 跳过 INT8 单测，bf16 upcast 路径无 CI 覆盖，纯测试改动、无运行时风险

关联脉络

- PR #33615 [Test] Route GEMM backend UTs through real layer modules and weight loaders: 本 PR 新增的 test_int8_linear_methods.py 完全沿用其 " 真实层 + 权重加载器 + oracle 对比 " 模式（make_tp1_column_parallel_linear、load_linear_weights、assert_output_close、test_fp8_blockwise_linear_backends.py），是同一测试基础设施演进线的下一环。
- PR #33619 [CI] Speed up dependency install: dual-ABI Rust ext cache and prevalidation pruning: 与本 PR 同属 " 削减 H100 per-commit CI 耗时 " 系列（head_ref lsyin/trim-h100-ci），一个优化依赖安装，一个裁剪测试矩阵。
- PR #33654 [CI] Move CPU-only unit tests to the CPU suite and trim dead 5090 registrations: 同为 CI 测试资产瘦身与注册整理，与本 PR 在 ci_register 套件调整上同脉络。