PR #26871 完整报告

sgl-project/sglang

Refactor EAGLE infer tests: shared fixture + kits + overlap matrix

合并时间：2026-06-01 18:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26871

执行摘要

- 一句话：EAGLE 推测测试重构为共享 Fixture+Kit
- 推荐动作：建议阅读本 PR 了解测试架构重构实践，尤其是 Fixture+Kit 模式如何提高可维护性和覆盖率表达能力。该模式值得在 SGLang 其他测试模块中推广。

功能与动机

原有测试文件命名无意义 (a/b/beta)，混杂引擎 / 服务器、模型和 overlap 模式，难以维护和扩展。PR 旨在通过共享 Fixture+Kit 模式提高可维护性，并显式固定后端以避免默认值更改导致测试失效。引用 PR body：'The old a/b/beta files mixed engine-vs-server, models, and overlap modes arbitrarily and the names carried no meaning.'

实现拆解

1. 创建统一启动底座 SpecEagleServerBase 在 python/sglang/test/server_fixtures/spec_eagle_fixture.py 中新增。通过类属性（如 spec_topk、page_size、attention_backend、disable_overlap）完全描述服务启动参数，子类只需覆写属性即可切换配置。setUpClass 根据属性构造命令行参数并启动子进程，tearDownClass 负责清理。
2. 构建可复用测试方法 Mixin（Kits） 在 python/sglang/test/kits/spec_server_kits.py 中新增。提供 SpecCorrectnessKit（接受长度、EOS 等）、SpecParityKit（无损输出对比）、SpecAccuracyKit（GSM8K 准确率）、SpecLogprobKit（logprob 无损）、SpecPenaltyKit（惩罚参数）、SpecFeatureKit（特性集合）。每个 Kit 是独立的 test_* 方法集合，无启动逻辑，阈值作为类属性可调。
3. 按特性x运行器拆分新测试文件 创建 7 个新测试文件，每个文件包含若干测试类，通过多继承组合底座和所需 Kit。例如： - test_spec_eagle.py —— 5090+flashinfer，测试 overlap x no-overlap 矩阵 - test_spec_eagle_topk.py —— topk>1 树形 draft - test_spec_eagle_page.py —— 不同 page 大小 - test_spec_eagle_triton.py —— triton 后端 - test_spec_eagle_fa3.py —— FA3 后端（H200） - test_spec_eagle_parity.py —— 无损输出 parity（H200） - test_spec_eagle_stress.py —— 性能、retract、超时等 所有新文件标注 stage='base-b'，在 CI 上运行；昂贵测试仅由 runner_config 区分。
4. 删除旧测试文件 test/registered/spec/eagle/test_eagle_infer_a.py、test_eagle_infer_b.py、test_eagle_infer_beta.py 被删除，因为它们的功能已被新文件覆盖（超集）。

5. 配套调整 - 修复顺序 parity 测试中参考服务的 OOM（提高 mem_fraction_static）。 - 修复 v1 batch_generation 的 KeyError（改用 meta_info）。 - 为 FR-Spec token map 和 EAGLE3 topk16 设置合理的 GSM8K 接受长度阈值。 - 确保 page64 后端从 triton 改为 flashinfer 后仍通过 logprob 无损门。

关键文件：

- python/sglang/test/kits/spec_server_kits.py（模块 测试 Kits；类别 test；类型 test-coverage；符号 SpecCorrectnessKit, test_acc_length, test_batch_generation, test_eos_token）：核心新增文件，包含所有可复用测试方法 Mixin（SpecCorrectnessKit 等），是整个测试架构的基石。
- python/sglang/test/server_fixtures/spec_eagle_fixture.py（模块 测试 Fixture；类别 test；类型 test-coverage；符号 SpecEagleServerBase, _launch_args, setUpClass, tearDownClass）：统一服务器启动底座，通过类属性完全描述配置，替代旧文件各自重复的启动逻辑。
- test/registered/spec/eagle/test_spec_eagle.py（模块 主测试；类别 test；类型 test-coverage；符号 _Core, TestEagle3Overlap, TestEagle3NoOverlap）：主测试文件，覆盖 overlap x no-overlap 矩阵，组合 _Core、SpecCorrectnessKit、SpecAccuracyKit 等。
- test/registered/spec/eagle/test_spec_eagle_topk.py（模块 TopK 测试；类别 test；类型 test-coverage；符号 TestEagle3Topk16, TestEagleLlama2Suite, TestEagleLlama2Chunked4, TestEagleLlama3TokenMap）：覆盖 topk>1 树形 draft，固定 flashinfer 后端，包含多个测试类。
- test/registered/spec/eagle/test_eagle_infer_b.py（模块 旧 Eagle 测试；类别 test；类型 deletion；符号 TestEAGLEServerBasic, setUpClass, test_request_abort, test_gsm8k）：被删除的旧测试文件之一，功能已完全迁移到新文件中。

关键符号：SpecCorrectnessKit.test_acc_length, SpecCorrectnessKit.test_batch_generation, SpecCorrectnessKit.test_eos_token, SpecParityKit.setUpClass, SpecEagleServerBase._launch_args, SpecEagleServerBase.setUpClass, TestEagle3Overlap, TestEagle3NoOverlap, TestEagle3Topk16, TestEagleLlama2Suite

关键源码片段

python/sglang/test/kits/spec_server_kits.py

核心新增文件，包含所有可复用测试方法 Mixin（SpecCorrectnessKit 等），是整个测试架构的基石。

```
# python/sglang/test/kits/spec_server_kits.py
# 可复用的测试方法 Mixin：SpecCorrectnessKit 提供接受质量和 EOS 检查

class SpecCorrectnessKit:
    """ Acceptance-quality + EOS checks (单服务器，轻量) """

    # 可覆盖的接受长度阈值（子类调整）
    acc_length_thres = 3.1
```

```
    batch_accept_len_thres = 1.75

    def test_acc_length(self):
        """ 单请求接受长度应高于阈值 """
        prompt = ["Human: Give me a fully functional FastAPI server. Show the python code.

Assistant:"] * 5
        sampling_params = {"temperature": 0, "max_new_tokens": 512}
        output = requests.post(
            self.base_url + "/generate",
            json={"text": prompt, "sampling_params": sampling_params},
        ).json()[0]
        meta = output["meta_info"]
        if "spec_verify_ct" in meta and meta["spec_verify_ct"] > 0:
            acc_length = meta["completion_tokens"] / meta["spec_verify_ct"]
        else:
            acc_length = 1.0
        print(f"{acc_length=:.4f}")
        self.assertGreater(acc_length, self.acc_length_thres)

    def test_batch_generation(self):
        """ 批量请求接受长度应高于阈值，不依赖 server_info """
        prompts = ["Hello, my name is", "The president of the United States is",
                   "The capital of France is", "The future of AI is"]
        results = requests.post(
            self.base_url + "/generate",
            json={"text": prompts, "sampling_params": {"temperature": 0, "max_new_tokens": 50}},
        ).json()
        total_completion, total_verify = 0, 0
        for r in results:
            self.assertIn("text", r, f"Server error: {r}")
            meta = r["meta_info"]
            total_completion += meta["completion_tokens"]
            total_verify += meta.get("spec_verify_ct", 0)
        if total_verify > 0:
            acc_length = total_completion / total_verify
            print(f"batch {acc_length=:.4f}")
            self.assertGreater(acc_length, self.batch_accept_len_thres)

    def test_eos_token(self):
        """ 生成结果不应包含 EOS token """
        prompt = "[INST] <<SYS>>\nYou are a helpful assistant.\n<</SYS>>\nToday is a sunny
        day and I like [/INST]"
        res = requests.post(
            self.base_url + "/generate",
            json={
                "text": prompt,
                "sampling_params": {"temperature": 0.1, "max_new_tokens": 1024, "skip_special_
                tokens": False},
```

```
            },
        ).json()
        output = res["text"]
        tokens = self.tokenizer.encode(output, truncation=False)
        self.assertNotIn(self.tokenizer.eos_token_id, tokens)

    def test_first_token_finish(self):
        """ 极短 max_new_tokens (1-3) 不崩溃 """
        prompts = [f"There are {i} apples on the table. How to divide them equally?" for i in
        range(8)]
        sampling_params = [{"temperature": 0, "max_new_tokens": random.randint(1, 3)} for _ in
        range(8)]
        results = requests.post(
            self.base_url + "/generate",
            json={"text": prompts, "sampling_params": sampling_params},
        ).json()
        for r in results:
            self.assertIn("text", r, f"Server error: {r}")
```

python/sglang/test/server_fixtures/spec_eagle_fixture.py

统一服务器启动底座，通过类属性完全描述配置，替代旧文件各自重复的启动逻辑。

```
# python/sglang/test/server_fixtures/spec_eagle_fixture.py
# 统一 EAGLE/EAGLE3 推测解码服务器测试底座

class SpecEagleServerBase(CustomTestCase):
    """ 通过类属性启动单个推测服务器 """

    # --- 模型及推测算法配置 ---
    model = DEFAULT_TARGET_MODEL_EAGLE3
    draft_model = DEFAULT_DRAFT_MODEL_EAGLE3
    spec_algo = "EAGLE3"
    spec_steps = 5
    spec_topk = 1
    spec_tokens = 6

    # --- 运行时配置 ---
    page_size = 1
    attention_backend = "flashinfer"
    disable_overlap = False # False -> spec v2 (overlap), True -> v1
    mem_fraction_static = 0.75
    max_running_requests = 8
    chunked_prefill_size = 128
    dtype = "float16"
    cuda_graph_max_bs = None
    trust_remote_code = True

    # --- 额外覆写 ---
    env_overrides = () # (env_var, value) 列表，仅在启动时生效
```

```
    extra_args = () # 其他命令行参数

    @classmethod
    def _launch_args(cls):
        """ 根据类属性组装服务器启动参数列表 """
        args = [
            "--speculative-algorithm", cls.spec_algo,
            "--speculative-draft-model-path", cls.draft_model,
            "--speculative-num-steps", str(cls.spec_steps),
            "--speculative-eagle-topk", str(cls.spec_topk),
            "--speculative-num-draft-tokens", str(cls.spec_tokens),
            "--page-size", str(cls.page_size),
            "--attention-backend", cls.attention_backend,
            "--mem-fraction-static", str(cls.mem_fraction_static),
            "--max-running-requests", str(cls.max_running_requests),
            "--chunked-prefill-size", str(cls.chunked_prefill_size),
            "--dtype", cls.dtype,
        ]
        if cls.disable_overlap:
            args.append("--disable-overlap-schedule")
        if cls.trust_remote_code:
            args.append("--trust-remote-code")
        if cls.cuda_graph_max_bs is not None:
            args += ["--cuda-graph-max-bs", str(cls.cuda_graph_max_bs)]
        args += [str(a) for a in cls.extra_args]
        return args

    @classmethod
    def setUpClass(cls):
        cls.base_url = DEFAULT_URL_FOR_TEST
        cls.target_model = cls.model
        cls._tokenizer = None
        with contextlib.ExitStack() as stack:
            stack.enter_context(envs.SGLANG_ENABLE_ASYNC_ASSERT.override(True))
            stack.enter_context(envs.SGLANG_ALLOW_OVERWRITE_LONGER_CONTEXT_LEN.
            override(True))
            for env_var, value in cls.env_overrides:
                stack.enter_context(env_var.override(value))
            cls.process = popen_launch_server(
                cls.model, cls.base_url,
                timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
                other_args=cls._launch_args(),
            )

    @classmethod
    def tearDownClass(cls):
        kill_process_tree(cls.process.pid, wait_timeout=60)

    @property
```

```
    def tokenizer(self):
        if type(self)._tokenizer is None:
            type(self)._tokenizer = get_tokenizer(self.model)
        return type(self)._tokenizer
```

评论区精华

在 PR body 和 Issue 评论中，作者 hnyls2002 详细列出了每类测试的 Kit 组合矩阵，并解释了设计决策：

- 测试架构选择：采用 Fixture+Kit 模式替代原有多重继承混用，使配置和测试方法解耦。
- Parity 顺序执行：为避免同时加载两个 8B 模型导致 OOM，参考服务先启动、关闭后再启动推测服务。
- 阈值调整：为 FR-Spec token map 和 EAGLE3 topk16 设定了较低的接受长度阈值 (2.5, 2.4)，因为 token map 会降低接受率。
- 配置覆盖：新配置集是 main 的超集，但有两点有意差异：FR-Spec token map 仅通过 GSM8K 接受长度守卫，而非完整引擎方法集；EAGLE3 page64 后端从 triton 改为 flashinfer 但仍保持 logprob 无损断言。
- 测试架构设计：Fixture+Kit 模式 vs 旧多重继承 (design): 采用 Fixture+Kit 模式，提高可维护性和可扩展性，并确保行为一致。

风险与影响

- 风险：主要风险：
  1. 阈值敏感性：接受长度阈值是 per-model 的，模型更新可能改变接受率导致测试假阳性 / 假阴性。
  2. 顺序 parity OOM：parity 测试顺序启动两个模型，虽然已调高内存参数，但不同 GPU 型号可能仍会 OOM。
  3. 配置覆盖遗漏：虽然声称是严格超集，但旧文件中某些特定组合（如特定 extra_args）可能未在新文件中映射。
  4. 测试阶段变化：旧文件有的标记为 extra-a 阶段（如 test_eagle_infer_a.py），新文件全部归入 base-b，可能改变 CI 执行时序。 - 影响： - 对用户：无直接影响。 - 对开发者：添加新推测配置测试只需继承 SpecEagleServerBase 并混入所需 Kit，大幅降低重复代码。 - 对系统：CI 测试文件增多但更精细，相同模型配置可能重复执行但总体负载可控。 - 对团队维护：测试架构可扩展，支持未来新后端或新模型快速接入。 - 风险标记：测试阈值敏感性，顺序 parity OOM 风险，配置超集确认

关联脉络

- PR #26870 Make unified tree SWA hicache tests faithful to write-through backup: 同属于测试架构重构，将旧测试文件拆分为共享组件，增强覆盖。