

PR #50330 完整报告

vllm-project/vllm

[CI] Organize speculative decoding E2E tests by coverage

合并时间: 2026-08-01 09:05

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/50330>

执行摘要

- 一句话: 拆分规格解码 E2E 测试, CI 改目录作用域并修 DSpark 量化
- 推荐动作: 值得精读。这是 " 目录即 CI 契约 " 的完整落地范例, 对测试组织、CI 配置和源码 bug 联动修复都有参考价值。重点看三点: spec_decode.yaml 的目录作用域写法、utils.py 中 compute_acceptance_rate/compute_acceptance_len 的增量统计设计 (支持逐 prompt 评估取平均)、以及 dspark/utils.py 里 quant_config 覆写与 #49133 的关联。若团队也在治理巨型 E2E 测试文件, 本 PR 的拆分顺序和 follow-up 边界控制值得借鉴。

功能与动机

PR body 明确说明目标: "Split the monolithic speculative-decoding E2E suite into feature directories that match CI ownership." 以及 "Make Buildkite run directory scopes instead of keyword/file allowlists, so tests added to an existing area are discovered by default." 同时 "Restore nightly DFlash and DSpark acceptance/accuracy coverage, using poolside/Laguna-XS-2.1-NVFP4 to keep Laguna practical on smaller GPUs"。作者还承认这是 "intentionally opinionated: directory boundaries are the CI contract.", 并明确三个高 footprint Eagle 案例与四 -GPU job 不回归。

实现拆解

1. 拆分巨型测试文件: 删除 tests/v1/e2e/spec_decode/test_spec_decode.py (-1679 行), 将用例按功能迁入 draft_model/test_draft_model.py、ngram_suffix/test_ngram_suffix.py、eagle/test_eagle_correctness.py、mtp/test_mtp.py、speculators/test_speculators.py 以及 acceptance_rates/{dflash,dspark,mtp_other} 等新文件; 每个文件保持原有用例语义, 风险是拆分过程中可能丢用例或破坏 import。
2. 收敛共享工具与 fixture: 新增 tests/v1/e2e/spec_decode/utils.py, 承载 get_test_prompts、greedy_sampling、stochastic_sampling、evaluate_llm_for_gsm8k、compute_acceptance_rate、compute_acceptance_len 等 helper; 新增 conftest.py 提供 sampling_config、model_name、reset_torch_dynamo 等跨目录 fixture; acceptance_rates/utils.py 再抽出 load_and_process_dataset、run_acceptance_length_eval, 供 DFlash 与 DSpark 复用。draft_model 里 ArgsTest dataclass 统一了正确性 / 接受率 / GSM8K 精度多个维度的参数描述, 后续加组合只需追加一条数据。

3. CI 改为目录作用域: .buildkite/test_areas/spec_decode.yaml 用 pytest -v -s v1/e2e/spec_decode/speculators/ 等目录级命令替代旧的 -k "speculators or mtp_correctness" 关键词过滤; engine.yaml 移除被迁移走的条目; .buildkite/test-amd.yaml 同步拆分命令, 避免 AMD job 遗漏。目录成为测试归属的唯一契约。
4. 恢复 DFlash/DSpark 覆盖: test_dflash.py 以 DFlashCorrectnessConfig 声明 Qwen3-8B 与 Laguna-XS-2.1-NVFP4 两组配置, 校验 GSM8K accuracy 与 compute_acceptance_len, 并保留 mrv1/mrv2 双 runner 参数化; test_dspark.py 恢复 Qwen3-4B-FP8 与 Gemma4 NVFP4 两个 probabilistic 采样路径的接受率 / 接受长度 / GSM8K 三维回归, 阈值按实测均值乘 0.9 抗 flaky。
5. 源码配套修复两处: vllm/model_executor/models/gemma4_dspark.py 在 DFlash 层初始化处补 self.causal = _dflash_layer_causal(config, layer_idx), 修复 #48167 之后的 causal 元数据回归; vllm/v1/worker/gpu/spec_decode/dspark/utils.py 的 load_dspark_model 中, 在 VllmConfig post-init 后覆写 draft_vllm_config.quant_config = get_draft_quant_config(vllm_config), 隔离草稿的量化配置。
6. 配套微调: 提交历史显示还修复了 large_gpu_mark 造成的误跳过、增加 speculators job 超时、从 model_runner_v2.yaml 移除 MTP 案例等, 属于拆分后让 CI 变绿的必要收尾。

关键文件:

- tests/v1/e2e/spec_decode/test_spec_decode.py (模块 解码测试; 类别 test; 类型 deletion; 符号 AsyncSchedulingNotEnabledError, _skip_if_insufficient_gpus_for_tp, get_test_prompts, get_instruct_coder_messages) : 1679 行巨型 E2E 测试文件整体删除, 是本 PR 拆分动作的源头; 所有用例按功能迁入子目录, 删除本身定义了新的测试组织基线。
- tests/v1/e2e/spec_decode/draft_model/test_draft_model.py (模块 草稿模型; 类别 test; 类型 test-coverage; 符号 AsyncSchedulingNotEnabledError, ArgsTest, get_messages, some_high_acceptance_metrics) : 本 PR 最大的新增测试文件 (+388 行), 集中验证 draft_model 方式的正确性、量化组合、TP>1 与 EngineArgs 配置独立性, ArgsTest dataclass 是后续扩展组合的模板。
- tests/v1/e2e/spec_decode/utils.py (模块 测试工具; 类别 test; 类型 test-coverage; 符号 _skip_if_insufficient_gpus_for_tp, get_test_prompts, get_instruct_coder_messages, greedy_sampling) : 共享测试工具的统一出口, compute_acceptance_rate 与 compute_acceptance_len 支持累计指标增量计算, 是 DFlash/DSpark/Medusa 所有接受率回归的基础。
- vllm/v1/worker/gpu/spec_decode/dspark/utils.py (模块 草稿加载; 类别 source; 类型 dependency-wiring) : 源码级修复的关键点: 在 VllmConfig post-init 后覆写草稿量化配置, 隔离 NVFP4 target 与 BF16 草稿, 是激活 Gemma4 DSpark 测试后暴露的真实 bug。
- vllm/model_executor/models/gemma4_dspark.py (模块 模型层; 类别 source; 类型 data-contract) : 修复 #48167 之后引入的 Gemma4 DSpark causal 元数据回归, 属于注意力 mask 语义变更, 是本次迁移激活测试后暴露的第二个真实 bug。
- .buildkite/test_areas/spec_decode.yaml (模块 CI 配置; 类别 config; 类型 configuration) : CI 契约的核心载体: 从文件级 allowlist 和关键词过滤改为目录级 pytest

命令，新增测试默认被收集，是本 PR 设计意图的直接体现。

- tests/v1/e2e/spec_decode/ngram_suffix/test_ngram_suffix.py (模块 匹配解码; 类别 test; 类型 test-coverage; 符号 disable_vllm_compile_cache_on_rocm, test_ngram_and_suffix_correctness, test_ngram_gpu_default_with_async_scheduling, test_suffix_decoding_acceptance) : n-gram 与 suffix 两类无模型草稿方式的正确性与接受率用例从原文件独立出来，并保留 ROCm 编译缓存处理的平台分支。
- tests/v1/e2e/spec_decode/acceptance_rates/dflash/test_dflash.py (模块 接受率回归; 类别 test; 类型 test-coverage; 符号 DFlashCorrectnessConfig, test_dflash_reference_acceptance_lengths, test_dflash_correctness) : 恢复 nightly DFlash 接受率与 GSM8K 准确率覆盖，新增 Laguna-XS-2.1-NVFP4 小显存组合，并保留 mrv1/mrv2 双 runner 参数化。
- tests/v1/e2e/spec_decode/acceptance_rates/dspark/test_dspark.py (模块 接受率回归; 类别 test; 类型 test-coverage; 符号 test_gemma4_dspark_correctness_and_acceptance_rate, dspark_config, test_dspark_correctness_and_acceptance_rate) : 恢复 nightly DSpark 接受率 / 接受长度 / GSM8K 三维覆盖，包含 Gemma4 NVFP4 与 Qwen3-4B-FP8 两条 probabilistic 采样路径。
- tests/v1/e2e/spec_decode/acceptance_rates/utils.py (模块 测试工具; 类别 test; 类型 test-coverage; 符号 load_and_process_dataset, run_acceptance_length_eval) : 接受率评估的共享驱动: load_and_process_dataset 支持 gsm8k/mt-bench/humaneval 多数数据集，run_acceptance_length_eval 逐 prompt 计算增量接受长度并断言均值。
- tests/v1/e2e/spec_decode/eagle/test_eagle_correctness.py (模块 草稿回归; 类别 test; 类型 test-coverage; 符号 test_eagle_correctness_light, test_eagle_correctness_medium) : EAGLE/EAGLE3 正确性用例独立成文件，保留 attn_backend 平台参数化与 Blackwell/transformers 版本 skip 条件; 三个高 footprint 案例被有意移除。
- tests/v1/e2e/spec_decode/mtp/test_mtp.py (模块 多词预测; 类别 test; 类型 test-coverage; 符号 test_mtp_correctness) : MTP 正确性用例独立并新增 Gemma4 E4B assistant 组合，同时补充了 hybrid 模型在 Model Runner V2 下的 skip 保护。
- .buildkite/test_areas/engine.yaml (模块 CI 配置; 类别 config; 类型 configuration) : 从 engine 面积移除被迁移走的 spec_decode 条目，保证 CI 面积不重复执行、职责清晰。

关键符号: test_draft_model_correctness, test_draft_model_engine_args_tensor_parallelism, compute_acceptance_rate, compute_acceptance_len, run_acceptance_length_eval, load_and_process_dataset, test_dflash_correctness, test_dflash_reference_acceptance_lengths, test_dspark_correctness_and_acceptance_rate, test_gemma4_dspark_correctness_and_acceptance_rate, test_ngram_and_suffix_correctness, test_suffix_decoding_acceptance, test_mtp_correctness, _run_eagle_correctness, test_eagle_correctness_light, test_eagle_correctness_medium, test_speculators_model_integration, load_dspark_model

关键源码片段

[tests/v1/e2e/spec_decode/draft_model/test_draft_model.py](#)

本 PR 最大的新增测试文件 (+388 行)，集中验证 draft_model 方式的正确性、量化组合、TP>1 与 EngineArgs 配置独立性，ArgsTest dataclass 是后续扩展组合的模板。

```
# tests/v1/e2e/spec_decode/draft_model/test_draft_model.py
# 拆分自原 test_spec_decode.py: 集中验证 draft_model 规格解码方式的正确性、
# 量化组合、TP>1 与 EngineArgs 配置独立性。
```

```
from dataclasses import dataclass
```

```
import pytest
```

```
from vllm import LLM, SamplingParams
from vllm.config import VllmConfig, replace
from vllm.engine.arg_utils import EngineArgs
```

```
from ..utils import (
    Messages,
    _skip_if_insufficient_gpus_for_tp,
    evaluate_llm_for_gsm8k,
    get_instruct_coder_messages,
    get_test_prompts,
    greedy_sampling,
    stochastic_sampling,
)
```

```
class AsyncSchedulingNotEnabledError(AssertionError):
    """draft_model 规格解码未启用 async_scheduling 时抛出。"""
```

```
@dataclass
```

```
class ArgsTest:
```

```
    """用一个 dataclass 统一描述一组规格解码验证参数。
```

```
    后续新增 draft_model 组合场景时,只需追加一条 ArgsTest,
    参数化会自动展开,避免为每种组合重复写引擎初始化样板。
    """
```

```
    target_model: str
    draft_model: str
    sampling_config: SamplingParams
    num_speculative_tokens: int
    expected_acceptance_rate: float
    expected_acceptance_len: float
    # 默认跳过 GSM8K 精度校验; 需要时显式给出阈值
    expected_gsm8k_accuracy: float = 0.0
    enforce_eager: bool = True
    parallel_drafting: bool = False
    target_tensor_parallel_size: int = 1
    draft_tensor_parallel_size: int = 1
```

```

max_model_len: int = 2048
gpu_memory_utilization: float = 0.5
dataset: str = "test_prompts"
num_prompts: int = 100

def get_messages(dataset: str, n: int) -> list[Messages]:
    """按数据集名字路由到不同的 prompt 构造器,测试数据源可插拔。"""
    if dataset == "test_prompts":
        return get_test_prompts(mm_enabled=False, num_prompts=n)
    if dataset == "likaixin/InstructCoder":
        return get_instruct_coder_messages(n=n)
    raise NotImplementedError(f"Dataset '{dataset}' not implemented")

# 覆盖两类代表性组合：同模型贪心（接受率上限）、小模型随机采样（真实收益）
cases = [
    ArgsTest(
        target_model="Qwen/Qwen3-0.6B",
        draft_model="Qwen/Qwen3-0.6B",
        sampling_config=greedy_sampling(),
        num_speculative_tokens=3,
        expected_acceptance_len=0.98 * (3 + 1), # K+1 的 epsilon 折扣
        expected_acceptance_rate=0.98,
        expected_gsm8k_accuracy=0.25,
    ),
    ArgsTest(
        target_model="Qwen/Qwen3-1.7B",
        draft_model="Qwen/Qwen3-0.6B",
        sampling_config=stochastic_sampling(),
        num_speculative_tokens=3,
        expected_acceptance_len=3.4,
        expected_acceptance_rate=0.80,
        expected_gsm8k_accuracy=0.5,
    ),
]

# draft_model 尚未启用 async_scheduling, 统一 xfail 到 issue #38929,
# 避免 CI 被已知短板拖红
@pytest.mark.parametrize("args", cases)
@pytest.mark.parametrize("enforce_eager", [True, False])
@pytest.mark.xfail(
    raises=AsyncSchedulingNotEnabledError,
    reason="draft_model does not yet enable async_scheduling: issue #38929",
)

def test_draft_model_correctness(args: ArgsTest, enforce_eager: bool):
    args.enforce_eager = enforce_eager
    assert_draft_model_correctness(args)

```

tests/v1/e2e/spec_decode/utils.py

共享测试工具的统一出口，compute_acceptance_rate 与 compute_acceptance_len 支持累计指标增量计算，是 DFlash/DSpark/Medusa 所有接受率回归的基础。

```
# tests/v1/e2e/spec_decode/utils.py
# 公共测试工具：原先散落在巨型测试文件里的 helper 全部收敛到这里，
# 供 draft_model、eagle、mtp、ngram_suffix 等子目录共享。
```

```
def compute_acceptance_rate(
    metrics: list[Metric], prev_metrics: list[Metric] | None = None
) -> float:
    """计算投机解码接受率(acceptance rate)。

    vLLM 暴露的指标是累计值,传入 prev_metrics 时计算两次采样之间的
    增量,适配逐条 prompt 评估后取平均的场景。
    """
    name2metric = {metric.name: metric for metric in metrics}
    n_draft_toks = name2metric["vllm:spec_decode_num_draft_tokens"].value
    if n_draft_toks == 0:
        return float("nan") # 无草稿 token 时无法定义接受率
    n_accepted_toks = name2metric["vllm:spec_decode_num_accepted_tokens"].value
    if prev_metrics is not None:
        prev_name2metric = {metric.name: metric for metric in prev_metrics}
        n_draft_toks -= prev_name2metric["vllm:spec_decode_num_draft_tokens"].value
        n_accepted_toks -= prev_name2metric[
            "vllm:spec_decode_num_accepted_tokens"
        ].value
    if n_draft_toks <= 0:
        return float("nan")
    return n_accepted_toks / n_draft_toks
```

```
def compute_acceptance_len(
    metrics: list[Metric], prev_metrics: list[Metric] | None = None
) -> float:
    """计算平均接受长度:期望每轮投机接受的 token 数,含第一个确定 token。"""
    name2metric = {metric.name: metric for metric in metrics}
    n_drafts = name2metric["vllm:spec_decode_num_drafts"].value
    n_accepted_toks = name2metric["vllm:spec_decode_num_accepted_tokens"].value
    if n_drafts == 0:
        return 1 # 没有投机轮次时退化为单 token 生成
    if prev_metrics is not None:
        prev_name2metric = {metric.name: metric for metric in prev_metrics}
        n_drafts -= prev_name2metric["vllm:spec_decode_num_drafts"].value
        n_accepted_toks -= prev_name2metric[
            "vllm:spec_decode_num_accepted_tokens"
        ].value
```

```
if n_drafts <= 0:
    return 1
return 1 + (n_accepted_toks / n_drafts)
```

vllm/v1/worker/gpu/spec_decode/dspark/utils.py

源码级修复的关键点：在 VllmConfig post-init 后覆写草稿量化配置，隔离 NVFP4 target 与 BF16 草稿，是激活 Gemma4 DSpark 测试后暴露的真实 bug。

```
# vllm/v1/worker/gpu/spec_decode/dspark/utils.py
# load_dspark_model 中的关键修复：草稿模型的量化配置必须与 target 隔离。

# VllmConfig 的 post-init 逻辑会把 target 的 quant_config 带回草稿配置，
# 而 target 配置需要保留用于 DSpark 的目标层元数据，因此这里必须
# 在构造完成后覆写一次草稿的量化配置，否则 NVFP4 target + BF16 草稿
# 这类组合会错误地按 NVFP4 加载草稿权重。
draft_vllm_config.quant_config = get_draft_quant_config(vllm_config)
```

评论区精华

评审人 AndreasKaratzas 在多个新测试文件 (dspark/test_dspark.py、medusa、synthetic、eagle/utils.py、mtp、ngram_suffix) 反复建议改用 vllm_runner 替代裸 LLM，理由是 "i would encourage vllm runners for this. This test does not run on AMD hardware, but trust me it will be more stable even on CUDA CI."; mgoin 回应: "Yes these are valid improvements but I'd like to scope this PR to just moving around existing tests and getting them working again. Improving or refactoring them should be a separate change". 关于 gemma4_dspark.py 的 causal 修复，mgoin 自述 "This is a genuine fix to Gemma4 DSpark since #48167 landed"; 对 load_dspark_model 的 quant_config 覆写，mgoin 表示 "I believe this will be a more general and simple solution to the problem #49133 tries to solve", 并注明 "dsv4 needs more changes, will reserve that for a followup". AndreasKaratzas 还建议 test-amd.yaml 增加 AMD mirror 镜像。最终其 APPROVED: "Let's merge this assuming that CI passes and work on the modifications that I proposed in a follow-up PR, to keep things containerized".

- 改用 vllm_runner 提升 E2E 稳定性 (testing): mgoin 认可为有效改进，但将本 PR 范围限定为 " 移动现有测试并恢复运行 "，统一改造留 follow-up。
- Gemma4 DSpark causal 元数据修复 (correctness): 确认合入该修复，由 Gemma4 DSpark E2E 用例长期守护。
- 草稿 quant_config 覆写与 #49133 的关系 (correctness): 本 PR 覆盖 Gemma4 NVFP4/BF16 场景，DSv4 NVFP4 后续处理。
- AMD 平台镜像与平台差异处理 (question): 平台兼容性改进未在本 PR 落实，留 follow-up；mgoin 表示保持本 PR 聚焦。
- 验收阈值、tqdm 与随机种子细节 (design): 保持现状合并，阈值保守化以抗 flaky；改进项列入 follow-up。

风险与影响

- 风险：
 - 大文件删除风险：1679 行文件整体删除，仓库内虽已清理 import，但外部 fork 或未搜索到的引用会直接断裂；此类重构合入后应尽快让全量 CI 跑一遍。
 - CI 目录作用域静默覆盖风险：Buildkite 改为按目录收集测试后，新测试若放错目录不会被 spec_decode job 发现，造成覆盖静默丢失；" 目录即契约 " 依赖命名纪律。
 - 模型语义修复风险：vllm/model_executor/models/gemma4_dspark.py 的 causal 赋值改变注意力 mask 语义，虽经 E2E 验证，但属于推理路径正确性变更，需 DSpark 测试长期守护。
 - 量化配置隔离不完整：vllm/v1/worker/gpu/spec_decode/dspark/utils.py 的覆写只验证了 Gemma4 NVFP4/BF16 场景，mgoin 自述 DSv4 NVFP4 仍需 follow-up，存在同路径其他模型组合未覆盖的隐患。
 - 阈值 flaky 风险：多个 acceptance rate/len 阈值是经验值乘 0.9 裕量，在 probabilistic 采样路径下仍可能因 GPU、驱动或模型下载差异抖动。
- 影响：
 - CI 基础设施：spec_decode.yaml、engine.yaml、test-amd.yaml 等多个面积从关键词匹配改为目录作用域，后续新增测试的文件位置直接决定 CI 归属，影响所有 spec_decode 相关开发者的提交流程。
 - 测试维护者：需要遵循新的目录约定，把用例放入对应 feature 目录；共享 utils.py 降低了跨用例的重复代码。
 - 产品功能：无用户可见功能变化；间接获得 DFlash、DSpark 与 Gemma4 NVFP4 DSpark 的持续回归保护。
 - 团队协作：9 个 commit 中多次 merge main、多次针对 CI 失败打补丁（超时、skip、平台差异），展示了 " 先拆分、再恢复绿灯 " 的渐进式迁移节奏；AI 辅助（OpenAI Codex）参与生成，作者声明已人工复核。
 - 风险标记：巨型测试文件拆分 (1679 行删除)，CI 目录作用域静默覆盖风险，模型语义修复 (causal 元数据)，DSpark NVFP4 仅覆盖部分场景，接受率阈值经验值可能 flaky

关联脉络

- PR #49133 DSpark quant dispatch 相关修复 (DeepSeek)：PR body 与 review 中 mgoin 均指出本 PR 在 load_dspark_model 中覆写 draft_quant_config 是该问题的更通用简单解法；本 PR 已覆盖 Gemma4 NVFP4/BF16 场景，DSv4 NVFP4 留待后续。
- PR #48167 Gemma4 DSpark 相关变更：mgoin 在 review 中说明 gemma4_dspark.py 的 self.causal 修复是针对 #48167 落地后引入的 causal 元数据回归。
- PR #38929 issue: draft_model 未启用 async_scheduling: 本 PR 大量 draft_model 用例统一 xfail 到该 issue；issue #38929 是 async_scheduling 能力补齐的追踪项，也是拆分后 CI 契约暴露的已知短板。
- PR #50242 K3 DSpark AR fusion: 同属 DSpark 投机解码加速演进线；本 PR 恢复了 DSpark 的 E2E 回归覆盖，与性能优化形成正确性守护闭环。