PR #50713 完整报告

vllm-project/vllm

[CI] Solidify speculative decoding E2E coverage

合并时间：2026-08-11 09:56

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50713

执行摘要

- 一句话：统一 spec-decode 测试基建，补 AMD CI 与失败诊断
- 推荐动作：值得精读。该 PR 是“大规模测试基建统一”的范本：vllm_runner 迁移中如何处理默认值语义差异（block_size/enable_chunked_prefill 显式传 None）、如何用共享工具函数收敛重复断言逻辑、以及如何在测试代码中做平台差异化（ROCm 的 AITER/FlashAttention 限制）都值得后续新增 spec-decode 测试时直接复用。重点文件是 tests/v1/e2e/spec_decode/utils.py 与 test_draft_model.py；若负责 CI 维护，还应关注 spec_decode.yaml 的 shard 平衡与新增 AMD 镜像。

功能与动机

PR body 明确说明这是 #50330 的 follow-up，目的是“apply some NITS across the new spec-decode area while hardening deterministic setup and actionable failure reporting”。结合代码变更可还原出三个具体动机：一是重组后的 spec-decode 测试仍以手写 LLM(...) + del llm + cleanup_dist_env_and_memory() 方式管理生命周期，各文件 setup、默认值、清理逻辑不一致（ROCm 上还需要 wait_for_rocm_memory_to_settle 等临时变通）；二是 prompt 生成依赖 random.seed(0)，匹配断言是各文件手写循环，失败时只输出零星 print、缺少可操作的诊断信息；三是原测试覆盖面偏 CUDA，MoE 后端断言硬编码 flashinfer 后端名导致 ROCm 无法运行，DFlash/DSpark 在 AMD CI 上也缺少镜像覆盖。

实现拆解

1. 统一测试生命周期到 vllm_runner：将 tests/v1/e2e/spec_decode 下 15+ 个测试文件（test_draft_model.py、test_lora.py、test_mtp_parallel_load.py、test_mtp.py、test_ngram_suffix.py、test_async.py、test_synthetic.py、test_dflash.py、test_dspark.py、test_speculators.py、test_medusa.py、eagle/utils.py、eagle/test_eagle_correctness.py、acceptance_rates/utils.py 等）从手写 LLM 构造 + 手动释放迁移到 with vllm_runner(...) as runner 上下文。vllm_runner 提供一致的清理与默认值管理，消除了 ROCm 上手工 wait_for_rocm_memory_to_settle 的必要。关键点在于 vllm_runner 与直接 LLM 构造的默认值不同（如 block_size 默认 16、enable_chunked_prefill 默认 False），因此多数调用点显式传入 block_size=None、enable_chunked_prefill=None、compilation_config=CompilationConfig() 以保持与 LLM 语义等价。

2. 抽取共享断言与指标工具：在 tests/v1/e2e/spec_decode/utils.py 新增 assert_request_outputs_match（带 required_matches 阈值的精确文本匹配，失败时输出最多 3 个 mismatch 的 text 与 token_ids 截断样本）和 get_spec_decode_metric_value（metric 缺失时列出可用 spec-decode 指标，给出可操作错误）。compute_acceptance_rate 与 compute_acceptance_len 转而基于该工具读取指标，消除了原先 KeyError 式失败。各测试文件（test_lora.py、test_mtp.py、eagle/utils.py 等）的手写匹配循环全部替换为 assert_request_outputs_match。
3. 平台感知与确定性处理：get_test_prompts 改用 set_random_seed(0) 同时覆盖 Python 与 Torch 随机状态；新增 _platform_moe_backend（ROCm 先设置 VLLM_ROCM_USE_AITER 与 VLLM_ROCM_USE_AITER_MOE 再返回 aiter，CUDA 返回 flashinfer_trtllm，其余返回 auto），使 MoE 后端断言可在 ROCm 上运行；新增 _enable_batch_invariance（ROCm 上显式关闭 VLLM_BATCH_INVARIANT，原因是 ROCm FlashAttention varlen API 缺少 num_splits 参数）；test_lora.py 的跳过条件从 is_cuda() 放宽为 is_cuda_alike()，CUBLAS_WORKSPACE_CONFIG 仅在 CUDA 上设置；EAGLE/MTP/dspark 测试通过 limit_mm_per_prompt（或 review 建议的 language_model_only）避免对未使用的多模态塔做 profiling。
4. 强化验收阈值与失败诊断：ArgsTest.num_prompts 从 100 提升到 200（注释称可将采样标准误降低约 29%）；匹配阈值从 matches > int(0.6 * len(ref_outputs)) 收紧为 required_matches=int(0.6 * len(ref_outputs)) + 1；pytest.raises(ValueError) 增加 match="draft_tensor_parallel_size" 精确匹配；test_mtp_parallel_load.py 对生成 token 数量与 MTP drafts 非零性增加断言（非空泛验证）；test_ngram_suffix.py 在断言失败时输出首末轮 accepted/drafted 摘要；acceptance_rates 相关测试移除 tqdm 进度输出。
5. CI 配置与配套：.buildkite/test_areas/spec_decode.yaml 新增 AMD CI 镜像（DFlash/DSpark，包括通过 ROCm 模拟的 NVFP4 目标）并平衡 Eagle shard；vllm/config/speculative.py 有 1 行微调（上下文未给出具体内容，属测试配置契约层面的配套调整）。

关键文件：

- tests/v1/e2e/spec_decode/utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 assert_request_outputs_match, get_spec_decode_metric_value, compute_acceptance_rate, compute_acceptance_len）：新增 assert_request_outputs_match 与 get_spec_decode_metric_value 两个共享工具，是整个测试加固的地基；compute_acceptance_rate / compute_acceptance_len 也重构为基于新工具，统一了失败诊断方式。
- 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 test_draft_model_correctness, test_draft_model_realistic_example, test_draft_model_parallel_drafting, test_draft_model_quantization）：draft_model 测试主文件：整体迁移到 vllm_runner，并将 MoE 后端断言平台化（_platform_moe_backend），使原本仅 CUDA 可跑的测试覆盖 ROCm；num_prompts 100→200 降低采样标准误。
- .buildkite/test_areas/spec_decode.yaml（模块 CI 配置；类别 config；类型 configuration）：CI 配置主体：新增 AMD DFlash/DSpark（含 NVFP4 ROCm 模拟）镜

像，平衡 Eagle shard，是本次 AMD 覆盖落地的关键配套。

- tests/v1/e2e/spec_decode/test_mtp_parallel_load.py（模块 MTP 加载；类别 test；类型 test-coverage；符号 _enable_batch_invariance, test_deepseek_mtp_load_inline, test_deepseek_mtp_load_dp）：MTP 并行加载测试：新增 _enable_batch_invariance 按平台显式开关 batch invariance（ROCm FlashAttention varlen API 缺 num_splits 参数），并迁移 vllm_runner；增加 token 数量与 drafts 非零断言防止空泛验证。
- tests/v1/e2e/spec_decode/draft_model/test_lora.py（模块 LoRA 测试；类别 test；类型 test-coverage；符号 test_batch_inference_correctness）：LoRA + spec-decode 测试：跳过条件从 is_cuda() 放宽为 is_cuda_alike() 使 ROCm 可跑，并替换为共享断言工具 assert_request_outputs_match，移除了手工匹配循环。

关键符号：assert_request_outputs_match, get_spec_decode_metric_value, compute_acceptance_rate, compute_acceptance_len, _platform_moe_backend, _enable_batch_invariance, test_draft_model_correctness, test_draft_model_moe_backend_override, test_draft_model_moe_backend_inherits_target, test_batch_inference_correctness, test_deepseek_mtp_load_inline, test_synthetic_acceptance_rate, _run_eagle_correctness

关键源码片段

tests/v1/e2e/spec_decode/utils.py

新增 assert_request_outputs_match 与 get_spec_decode_metric_value 两个共享工具，是整个测试加固的地基；compute_acceptance_rate / compute_acceptance_len 也重构为基于新工具，统一了失败诊断方式。

```
# 带阈值的精确文本匹配断言，失败时输出有界的诊断信息，
# 用于对比参考引擎与 spec-decode 引擎的输出是否一致。
def assert_request_outputs_match(
    ref_outputs: Sequence[RequestOutput],
    spec_outputs: Sequence[RequestOutput],
    *,
    required_matches: int,
    context: str,
    max_mismatches: int = 3,
) -> None:
    assert ref_outputs, f"{context}: no reference outputs"
    assert len(ref_outputs) == len(spec_outputs), (
        f"{context}: output count differs: "
        f"reference={len(ref_outputs)}, speculative={len(spec_outputs)}"
    )
    assert 0 <= required_matches <= len(ref_outputs), (
        f"{context}: invalid required_matches={required_matches} for "
        f"{len(ref_outputs)} outputs"
    )

    mismatches: list[str] = []
    matches = 0
```

```
    for index, (ref_output, spec_output) in enumerate(zip(ref_outputs, spec_outputs)):
        assert ref_output.outputs, (
            f"{context}: reference output {index} has no candidate"
        )
        assert spec_output.outputs, (
            f"{context}: speculative output {index} has no candidate"
        )
        ref_candidate = ref_output.outputs[0]
        spec_candidate = spec_output.outputs[0]
        if ref_candidate.text == spec_candidate.text:
            matches += 1
        elif len(mismatches) < max_mismatches:
            # 只记录前 max_mismatches 个不匹配，避免失败日志过长；
            # 同时截断文本与 token id，足够定位漂移点。
            mismatches.append(
                f"[{index}] ref_text={ref_candidate.text[:240]!r}, "
                f"spec_text={spec_candidate.text[:240]!r}\n"
                f"    ref_token_ids={list(ref_candidate.token_ids)[:64]}\n"
                f"    spec_token_ids={list(spec_candidate.token_ids)[:64]}"
            )

    print(
        f"{context}: exact text matches={matches}/{len(ref_outputs)} "
        f"(required={required_matches})"
    )
    mismatch_summary = "\n".join(mismatches) or "no mismatches captured"
    assert matches >= required_matches, (
        f"{context}: only {matches}/{len(ref_outputs)} outputs matched; "
        f"required at least {required_matches}. First mismatches:\n"
        f"{mismatch_summary}"
    )


# 读取 spec-decode 统计指标，缺失时给出可操作的错误信息，
# 避免原先的 KeyError 式失败。
def get_spec_decode_metric_value(metrics: Sequence[Metric], metric_name: str) -> float:
    name2metric = {metric.name: metric for metric in metrics}
    metric = name2metric.get(metric_name)
    assert metric is not None, (
        f"Missing metric {metric_name!r}. Ensure disable_log_stats=False. "
        "Available spec-decode metrics: "
        f"{sorted(name for name in name2metric if 'spec_decode' in name) or ['<none>']}"
    )
    return float(metric.value)
```

tests/v1/e2e/spec_decode/draft_model/test_draft_model.py

draft_model 测试主文件：整体迁移到 vllm_runner，并将 MoE 后端断言平台化（_platform_moe_backend），使原本仅 CUDA 可跑的测试覆盖 ROCm；num_prompts

100→200 降低采样标准误。

```
# 按当前平台选择可用的 MoE 后端名，并完成 ROCm 所需的
# 环境变量设置，保证后端相关断言在 ROCm CI 上也能运行。
def _platform_moe_backend(monkeypatch: pytest.MonkeyPatch) -> MoEBackend:
    if current_platform.is_rocm():
        monkeypatch.setenv("VLLM_ROCM_USE_AITER", "1")
        monkeypatch.setenv("VLLM_ROCM_USE_AITER_MOE", "1")
        return "aiter"
    if current_platform.is_cuda():
        return "flashinfer_trtllm"
    return "auto"


# 当 speculative_config 显式指定 moe_backend 时，草稿配置应使用它，
# 而目标模型保留自己的后端设置，互不影响。
def test_draft_model_moe_backend_override(monkeypatch: pytest.MonkeyPatch):
    target_moe_backend = _platform_moe_backend(monkeypatch)
    engine_args = EngineArgs(
        model="Qwen/Qwen3-1.7B",
        tensor_parallel_size=1,
        moe_backend=target_moe_backend, # 目标侧后端按平台选择
        speculative_config={
            "model": "Qwen/Qwen3-0.6B",
            "method": "draft_model",
            "num_speculative_tokens": 3,
            "moe_backend": "triton", # 草稿侧显式指定
        },
    )
    tgt_config: VllmConfig = engine_args.create_engine_config()
    assert tgt_config.kernel_config.moe_backend == target_moe_backend
    assert tgt_config.speculative_config.moe_backend == "triton"
    draft_config = _apply_draft_moe_backend(tgt_config)
    assert draft_config.kernel_config.moe_backend == "triton"
    # 应用草稿配置后，目标配置必须保持原状不受影响。
    assert tgt_config.kernel_config.moe_backend == target_moe_backend
```

tests/v1/e2e/spec_decode/test_mtp_parallel_load.py

MTP 并行加载测试：新增 _enable_batch_invariance 按平台显式开关 batch invariance（ROCm FlashAttention varlen API 缺 num_splits 参数），并迁移 vllm_runner；增加 token 数量与 drafts 非零断言防止空泛验证。

```
# DeepSeek MLA prefill 的 batch invariance 依赖 FlashAttention 的
# num_splits 参数，ROCm 上游 varlen API 不支持该参数，因此按平台
# 显式开关，避免环境变量被外部设置污染测试结果。
def _enable_batch_invariance(monkeypatch: pytest.MonkeyPatch) -> None:
    if current_platform.is_cuda():
        monkeypatch.setenv("VLLM_BATCH_INVARIANT", "1")
    else:
```

monkeypatch.setenv("VLLM_BATCH_INVARIANT", "0")

评论区精华

Review 由 mgoin 主导并最终 APPROVED，核心讨论集中在 dspark 测试的 VllmRunner 迁移参数上：

1) mgoin 质疑为何要显式设置 block_size 与 enable_chunked_prefill（认为默认值即可），作者解释 VllmRunner 与直接 LLM 构造的默认值不同（block_size=16、enable_chunked_prefill=False），移除会改变行为并禁用 Gemma4 支持的 chunked prefill，故必须保留显式传参以保持语义等价； 2) mgoin 提出 limit_mm_per_prompt 的写法可以用 language_model_only=True 简化，作者采纳。整体无未解决的争议。

- VllmRunner 默认值与 LLM 不一致的语义风险 (question): 保留显式传参以保持与 LLM 构造语义等价，这是迁移时必须记录的关键约定。
- limit_mm_per_prompt 可替换为 language_model_only (design): 采纳建议，使用更简洁的 language_model_only 表达。

风险与影响

- 风险：
  1. 大范围迁移的语义漂移风险：VllmRunner 与直接 LLM 构造的默认值不同（block_size、enable_chunked_prefill、trust_remote_code 等），虽然本 PR 多数入口显式传参，但 15+ 文件迁移中任何遗漏都会静默改变测试行为，可能造成假通过或假失败，后续维护需警惕。
  2. 平台分支增多：_platform_moe_backend、_enable_batch_invariance 等函数在 CUDA/ROCm 上行为不同，未来新平台（如 XPU）会落入 else 分支，语义可能不达预期。
  3. 阈值与统计收紧可能引发 flaky：num_prompts 100→200 增加测试时长；required_matches 从 >x 改为 >=floor(x)+1；synthetic acceptance 的 tolerance 0.15 在 ROCm 上可能因数值差异偶发失败。
  4. CI 资源与外部依赖：spec_decode.yaml 新增 AMD 镜像后 CI 时间与 GPU 资源占用上升；测试依赖 HuggingFace 外网模型下载，存在网络抖动导致的偶发失败。 - 影响：影响范围集中在测试与 CI：tests/v1/e2e/spec_decode 下所有测试的写法成为后续新增 spec-decode 测试的样板；AMD ROCm CI 首次获得 DFlash/DSpark 及 NVFP4 模拟覆盖，跨平台回归检测能力显著增强。对生产推理路径无影响（vllm/config/speculative.py 仅 1 行改动）。对团队而言，失败诊断从“零星 print + 手写阈值”升级为“带上下文与样本截断的结构化断言”，定位 flaky 的成本降低。对用户无行为影响。 - 风险标记：测试基建全量迁移，VllmRunner 默认值差异带来语义变化，阈值收紧可能引发跨平台 flaky，CI 资源与镜像依赖增加

关联脉络

- PR #50330 （前序 PR，标题未在上下文中提供）：PR body 明确说明本 PR 是 #50330 的 follow-up，对重组后的 spec-decode 测试区域做 NITS 加固。

- PR #50693 Fix DSpark warmup without sparse index buffer: 同一 spec-decode 测试区域（DSpark）的 bugfix，本 PR 同步为 DSpark 测试增加 AMD CI 覆盖与平台适配。
- PR #47352 [Model Runner V2][MTP] Share topk index buffer between draft steps: MTP 相关演进，本 PR 的 test_mtp_parallel_load.py 与 test_mtp.py 均涉及 MTP 并行加载与平台化处理。