PR #36032 完整报告

sgl-project/sglang

test: the 5090 consumer case runs two warm requests on the full recipe

合并时间：2026-08-23 15:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36032

执行摘要

- 一句话：5090 MiniMax-H3 用例连发两次 warm 请求并收紧基线
- 推荐动作：值得快速阅读：它展示了如何用最小改动（一个字段 + 一层循环）把 " 单请求正确 " 升级为 " 跨请求无泄漏 " 的 CI 防线，测试设计思路可复制到其他长生命周期服务场景。建议重点看 _test_diffusion_generation_impl 的循环改法和 _make_5090_h3_consumer_budget_case 中通过环境变量模拟受限硬件的技巧。若团队后续修改 layerwise offload 或 VAE residency 逻辑，这个用例就是第一道回归防线。

功能与动机

PR body 明确指出："State leaking across requests (residency arming, courier in-flight tracking, host copies) shows up as the warm request degrading or dying, which a single-request case cannot see." 单次请求的用例无法发现跨请求的状态泄漏，只有同会话内的第二次 warm 请求才能让泄漏表现为性能降级或 OOM。此外，cookbook recipe 在 #35967 之后已改为 video_vae=36（解码器权重以 decode dtype 加载），测试用例需要跟随 recipe 才能守护真实部署配置。

实现拆解

1. 新增重复请求开关：在 python/sglang/multimodal_gen/test/server/testcase_configs.py 的 DiffusionTestCase dataclass 中新增 perf_repeat_requests: int = 1 字段，默认值保持单次请求行为，不影响其他用例。
2. 改造生成执行流程：在 python/sglang/multimodal_gen/test/server/test_server_common.py 的 _test_diffusion_generation_impl 中，把单次 run_and_collect 调用改为 for _ in range(max(1, case.perf_repeat_requests)) 循环，最后一次请求的 perf_record 与 content 用于后续性能与一致性校验；GT 生成模式仍只保存最后一次输出，行为不变。
3. 更新 5090 consumer 用例：在 python/sglang/multimodal_gen/test/server/gpu_cases.py 的 _make_5090_h3_consumer_budget_case 中，把 --layerwise-resident-layers 从 video_vae=24 提升到 video_vae=36（对齐 cookbook recipe 现状），并设置 perf_repeat_requests=2，让第二次 warm 请求与第一次走同一生成管线。
4. 收紧性能基线：在 python/sglang/multimodal_gen/test/server/perf_baselines/5090.json 中，MiniMaxH3DecodingStage 从 26550 ms 收紧到 13000 ms，expected_e2e_ms 从 125000 ms 收紧到 105000 ms，estimated_full_test_time_s 从 900 s 调整为 1050 s（多跑一次请求带来的时长增加）。

5. 离 CI 验证：PR body 说明已在匹配环境验证两次顺序与两次并发请求下 GPU/host 内存保持平稳、无 OOM，证明 warm-request 校验在该用例上可行。

关键文件：

- python/sglang/multimodal_gen/test/server/test_server_common.py（模块 服务端测试；类别 test；类型 test-coverage；符号 _test_diffusion_generation_impl）：生成执行流程的核心改动点：从单次 run_and_collect 改为按 perf_repeat_requests 循环，第二次 warm 请求的性能与输出成为校验对象，是本次能力的执行核心。
- python/sglang/multimodal_gen/test/server/testcase_configs.py（模块 用例配置；类别 test；类型 test-coverage；符号 DiffusionTestCase）：为 DiffusionTestCase 新增 perf_repeat_requests 字段，是所有用例共享的重复请求开关，默认值 1 保证既有用例行为不变。
- python/sglang/multimodal_gen/test/server/gpu_cases.py（模块 GPU 用例；类别 test；类型 test-coverage；符号 _make_5090_h3_consumer_budget_case）：5090 消费级用例的落地配置：video_vae 从 24 提升到 36 对齐 cookbook recipe，并开启 perf_repeat_requests=2 启用 warm 请求校验。
- python/sglang/multimodal_gen/test/server/perf_baselines/5090.json（模块 性能基线；类别 test；类型 test-coverage）：性能基线随 fp16 解码器与双次请求调整：*decode* 从 26550 ms 收紧到 13000 ms，*e2e* 从 125000 ms 收紧到 105000 ms，预估时长从 900 s 升到 1050 s。

关键符号：_test_diffusion_generation_impl, _make_5090_h3_consumer_budget_case

关键源码片段

python/sglang/multimodal_gen/test/server/test_server_common.py

生成执行流程的核心改动点：从单次 run_and_collect 改为按 perf_repeat_requests 循环，第二次 warm 请求的性能与输出成为校验对象，是本次能力的执行核心。

```
# 在同一个 server session 内把请求连发 perf_repeat_requests 次，
# 最后一次请求的输出同时用于性能校验与一致性校验。
# 当 perf_repeat_requests > 1 时，追加的请求相当于 warm 请求：
# 若 residency 预载、courier 在途跟踪或 host 拷贝等状态在请求间泄漏，
# warm 请求会表现为性能降级甚至 OOM，从而被基线校验捕获。
is_realtime_case = case.sampling_params.realtime_num_chunks is not None
for _ in range(max(1, case.perf_repeat_requests)):
    perf_record, content = self.run_and_collect(
        diffusion_server,
        case.id,
        generate_fn,
        collect_perf=not is_gt_gen_mode and not is_realtime_case,
    )

# 循环结束后，perf_record 与 content 属于最后一次请求；
# 后续的 realtime 分块统计或常规性能校验都以这份数据为准。
if is_realtime_case:
```

```
    chunk_stats = pop_realtime_perf_stats(case.id)
```

python/sglang/multimodal_gen/test/server/testcase_configs.py

为 DiffusionTestCase 新增 perf_repeat_requests 字段，是所有用例共享的重复请求开关，默认值 1 保证既有用例行为不变。

```
@dataclass(frozen=True)
class DiffusionTestCase:
    """单个 model/scenario 测试用例的配置结构。"""

    id: str # pytest test id 兼场景名
    server_args: DiffusionServerArgs
    sampling_params: DiffusionSamplingParams | None = None
    run_perf_check: bool = True

    # 在同一 server session 内发送请求的次数，性能与一致性只在最后一次请求上校验。
    # 取值大于 1 时，用于断言 warm 的第二次请求也能满足第一次请求的同一套基线；
    # 这样 residency 预载、courier 在途跟踪、host 拷贝上的跨请求状态泄漏，
    # 就会表现为第二次请求性能降级或失败，而不是被单次请求掩盖。
    perf_repeat_requests: int = 1

    run_consistency_check: bool = True
    run_component_accuracy_check: bool = True
    run_models_api_check: bool = True
    run_t2v_input_reference_check: bool = True
    run_lora_basic_api_check: bool = False
    run_lora_dynamic_load_check: bool = False
    run_lora_dynamic_switch_check: bool = False
    run_multi_lora_api_check: bool = False
```

评论区精华

本 PR 没有任何 review 评论或讨论线程（comments_count=0、review_comments_count=0），由作者自行合并。最有价值的信息来自 PR body 的离 CI 验证结论：两次顺序与两次并发请求下 GPU/host 内存均保持平稳、无 OOM，为 warm-request 校验的有效性提供了直接依据。另一个值得注意的设计自述是：该用例通过环境变量 SGLANG_DIFFUSION_TEST_FORCE_HOST_AVAILABLE_GIB=32 与 SGLANG_DIFFUSION_TEST_CAP_DEVICE_MEMORY_GIB=12 模拟 12 GiB 显卡 + 32 GiB 主机的消费级环境，让真实硬件走受限放置路径，从而把 "fits 12 GiB" 从推断变成强制约束。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. CI 时长增加：estimated_full_test_time_s 从 900 s 升到 1050 s（约 +17%），5090 diffusion CI 车道会变慢，属于可接受的换防成本。

2. 基线收紧依赖 fp16 解码器行为：*decode* 基线从 26550 ms 腰斩到 13000 ms，*e2e* 从 125000 ms 收紧到 105000 ms，若未来解码器 *dtype* 或加载方式回退，该用例会立刻误报失败；但这也是本 PR 想守护的回归点。
3. warm 请求校验依赖同会话复用：*perf_repeat_requests* 的语义建立在同一个 server session 内连发请求之上，若 *run_and_collect* 或外层 fixture 在不同迭代间重启 server，校验将退化为两次冷启动对比、失去检测泄漏的意义；当前片段显示循环在 diffusion_server 上下文内复用，符合预期。
4. 测试代码对运行时行为的隐式耦合：测试通过模拟环境变量强制走 offload 路径，若运行时改掉这些 *env var* 的接线，用例可能静默失去覆盖力。 - 影响：影响范围限定在 diffusion 测试体系：5090 MiniMax-H3 *consumer-budget* 用例的执行时长增加约 150 s，并对跨请求状态泄漏新增了明确防线；DiffusionTestCase 新增的 *perf_repeat_requests* 字段对所有用例开放，未来其他 GPU 或模型用例也可复用 warm-request 校验模式。对运行时、API 和用户无任何影响。对团队的实质收益是：*residency* 预载、*courier* 在途跟踪、host 拷贝这类容易在单请求下漏检的状态管理 bug，现在会被 CI 自动捕获。 - 风险标记：CI 时长增加，基线收紧依赖 fp16 解码器，同会话 warm 请求依赖正确复用

关联脉络

- PR #35967 *decoder weights load in their decode dtype*: PR body 明确引用：#35967 让解码器权重以 *decode dtype* 加载，*cookbook recipe* 随之改为 video_vae=36，本 PR 的用例配置跟随该变更。
- PR #36051 [diffusion] CI: *guard the anonymous-host budget alongside peak* VRAM: 同一测试套件的 CI 加固系列：同为 diffusion 测试基础设施改动，且都修改了 test_server_common.py、testcase_configs.py、perf_baselines/5090.json，与本 PR 在文件层面直接重叠。
- PR #36034 [diffusion] UX: *clean up startup and offload logs*: 涉及 layerwise_offload 与 *memory manager* 的改动，属于本 PR 要守护的跨请求状态泄漏域（residency、courier、host 拷贝），后续回归可由本用例捕获。
- PR #35813 [diffusion] *stop the mapped-weight store from holding the parameter itself*: layerwise_offload 映射权重存储的 bugfix，与 warm 请求关注的 *courier* 在途跟踪和 host 拷贝状态同属 offload 状态管理，本用例可提供跨请求回归覆盖。