PR #32568 完整报告

sgl-project/sglang

[AMD] Add Kimi-K3 8-GPU MI35x nightly accuracy CI

合并时间：2026-08-18 05:48

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32568

执行摘要

- 一句话：MI35x 新增 Kimi-K3 精度 nightly，替换 K2.6
- 推荐动作：值得精读，尤其是 6 个 commit 的演进过程。这个 PR 虽无源码逻辑，但展示了高质量的 CI 工程判断：精度先于性能、用硬件 ISA 约束校验 " 替换 " 决策、不 pin 废弃镜像、用 completion harness 规避 chat 模板陷阱。对想了解 sglang AMD nightly 编排（register_amd_ci、run_suite.py、job_select 联动）的读者是很好的范本。

功能与动机

PR body 明确指出：Kimi-K3 cookbook 虽然发布了 MI350X/MI355X 1×8 部署配方，AMD Day-0 跟踪 issue #32548 也报告了吞吐，但两者都没有提供自动化端到端精度信号——cookbook 该 cell 仍标记为 verified: false，#32548 只测速度。测试文件 docstring 进一步说明："That issue reports throughput on MI355 TP8 but no accuracy, and the cookbook cell for this topology is still published as verified: false. So the recipe is known to run at speed; what is missing, and what this test supplies, is evidence that it produces correct tokens." 即配方能跑多快已知，缺的是 " 跑得对不对 " 的证据。

实现拆解

实现分四步完成，核心是新增一个端到端精度测试并接入 AMD nightly 编排：

1. 新增 MI35x 精度测试文件test/registered/amd/accuracy/mi35x/test_kimi_k3_eval_mi35x.py（+163 行）。文件顶部调用 register_amd_ci(est_time=9000, suite="nightly-amd-accuracy-8-gpu-mi35x-kimi-k3", nightly=True) 完成 suite 注册；TestKimiK3EvalMI35x.setUpClass 通过环境变量 GSM8K_NUM_QUESTIONS、GSM8K_MAX_NEW_TOKENS 提供可调入口；test_kimi_k3_gsm8k_accuracy 用 popen_launch_server 拉起 TP8 服务，参数完全对齐 #32548 的非投机配方（Triton attention、bfloat16、--mem-fraction-static 0.85、--disable-radix-cache），并设置 --reasoning-parser kimi_k3、--tool-call-parser kimi_k3。关键设计是选用 few-shot completion 评估而不是 chat 路径：K3 的 thinking 永久开启，答案走 reasoning_content，chat 路径的 message.content 为空会得 0 分；completion 路径直接对原始输出打分，并与既有 Kimi 兄弟测试（K2/K2.5/K2.6）保持一致。环境方面设置了 4 个 AITER 相关变量（SGLANG_USE_AITER、SGLANG_AITER_K3_OPT、AITER_FLYDSL_FORCE、AITER_SITUV2_A8W4），其中前两个在 sglang 内被读取，后两个由 AITER 自身消费。评估完成后按 ACCURACY_THRESHOLD = 0.92 断言（与 AMD Kimi 精度家族一致），

CI 下将结果写入 GitHub step summary，finally 中 kill_process_tree 确保 server 进程被回收。

2. 工作流替换.github/workflows/nightly-test-amd-rocm720.yml（+21/-10）。将 nightly-8-gpu-mi35x-kimi-k26-rocm720 替换为 nightly-8-gpu-mi35x-kimi-k3-rocm720，同步更新 job_select 与 check-all-jobs 两处条目；MI30x 的 nightly-8-gpu-kimi-k26-rocm720 保留，注释从 "MI30x + MI35x" 改为 "MI30x"。由于 2.8T 权重的 MXFP4 checkpoint 约 1.56 TB，加载耗时占大头，job 超时从 180 分钟提升到 300 分钟，--timeout-per-file 从 7200 升到 14400。
3. suite 注册test/run_suite.py（+1/-0）。在 NIGHTLY_SUITES[HWBackend.AMD] 列表中插入 nightly-amd-accuracy-8-gpu-mi35x-kimi-k3，使 run_suite.py --hw amd --suite ... 能发现该测试。
4. 提交演进中的范围收敛（来自 6 个 commit）。初始版本同时包含 accuracy 与 perf 两个 job，且 perf 通过 needs: 依赖 accuracy 通过后才运行；但 perf job 在唯一一次分派中 23 秒即死于 setUpClass（基于 #33832 之前的 NightlyBenchmarkRunner API 编写），因此被拆出推迟到 #34985。随后恢复 MI300 的 K2.6 任务、保留 ROCm 7.0 工作流不动，最终只替换 7.2 lane 的 MI35x 槽位。

关键文件：

- test/registered/amd/accuracy/mi35x/test_kimi_k3_eval_mi35x.py（模块 测试用例；类别 test；类型 test-coverage；符号 TestKimiK3EvalMI35x, setUpClass, test_kimi_k3_gsm8k_accuracy）：核心新增文件，定义了 Kimi-K3 在 MI35x 8 卡上的 GSM8K 精度测试：服务器参数、AITER 环境变量、评估断言与 CI summary 输出全部集中于此。
- .github/workflows/nightly-test-amd-rocm720.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：工作流编排的替换点：将 MI35x 8 卡槽位从 K2.6 换成 K3，并调整超时、job_select、check-all-jobs 等配套条目。
- test/run_suite.py（模块 测试编排；类别 test；类型 test-coverage）：suite 注册入口，将新测试接入 NIGHTLY_SUITES[AMD]，使 run_suite.py 能按 suite 名发现并运行该 test。

关键符号：TestKimiK3EvalMI35x.setUpClass, TestKimiK3EvalMI35x.test_kimi_k3_gsm8k_accuracy

关键源码片段

test/registered/amd/accuracy/mi35x/test_kimi_k3_eval_mi35x.py

核心新增文件，定义了 Kimi-K3 在 MI35x 8 卡上的 GSM8K 精度测试：服务器参数、AITER 环境变量、评估断言与 CI summary 输出全部集中于此。

```
class TestKimiK3EvalMI35x(CustomTestCase):
    """Kimi-K3 GSM8K 精度评估测试，运行在 AMD MI35x 8 卡（TP8）上。"""

    @classmethod
    def setUpClass(cls):
        cls.base_url = DEFAULT_URL_FOR_TEST
        # 允许通过环境变量覆盖题目数与生成长度，便于本地快速调试
```

```
    cls.num_questions = int(os.environ.get("GSM8K_NUM_QUESTIONS", "1319"))
    cls.max_new_tokens = int(os.environ.get("GSM8K_MAX_NEW_TOKENS", "512"))

def test_kimi_k3_gsm8k_accuracy(self):
    # 服务器参数对齐 #32548 非投机配方：TP8 + Triton attention +
    # bfloat16 + 静态内存占比 0.85，并关闭 radix cache
    other_args = [
        "--tp", str(TP_SIZE),
        "--attention-backend", "triton",
        "--dtype", "bfloat16",
        "--mem-fraction-static", "0.85",
        "--disable-radix-cache",
        # 并发上限与 decode graph 捕获上限保持一致：1319 题一次性
        # 全量提交，必须显式限流，避免突破权重之后仅剩的约 53 GB/ 卡
        # 显存预算；捕获超过并发上限的 decode graph 没有收益
        "--cuda-graph-max-bs-decode", str(MAX_RUNNING_REQUESTS),
        "--max-running-requests", str(MAX_RUNNING_REQUESTS),
        # K3 的 thinking 永远开启，需用专用 parser 解析输出
        "--reasoning-parser", "kimi_k3",
        "--tool-call-parser", "kimi_k3",
        "--trust-remote-code",
        '--model-loader-extra-config', '{"enable_multithread_load": true}',
        "--watchdog-timeout", "1200",
    ]

    env = os.environ.copy()
    # K3 的 MoE 在 ROCm 上由 AITER 提供，四个环境变量缺一不可：
    # SGLANG_USE_AITER 是总开关；SGLANG_AITER_K3_OPT 在 mxfp4.py 与
    # models/kimi_k3.py 中被读取；AITER_SITUV2_A8W4 选择 W4A8 SiTU
    # expert 内核；AITER_FLYDSL_FORCE 由 AITER 自身消费，仓库内不 grep
    env["SGLANG_USE_AITER"] = "1"
    env["SGLANG_AITER_K3_OPT"] = "1"
    env["AITER_FLYDSL_FORCE"] = "1"
    env["AITER_SITUV2_A8W4"] = "1"

    process = popen_launch_server(
        KIMI_K3_MODEL_PATH, self.base_url,
        timeout=SERVER_LAUNCH_TIMEOUT, other_args=other_args, env=env,
    )

    try:
        requests.get(self.base_url + "/flush_cache")
        args = SimpleNamespace(
            num_shots=8, data_path=None,
            num_questions=self.num_questions,
            parallel=self.num_questions,
            max_new_tokens=self.max_new_tokens,
            host="http://127.0.0.1",
            port=int(self.base_url.split(":")[-1]),
```

```
            )
            metrics = run_eval_few_shot_gsm8k(args)
            acc = metrics["accuracy"]
            passed = acc >= ACCURACY_THRESHOLD
            status = "✅ PASS" if passed else "❌ FAIL"
            print(f"  accuracy={acc:.3f} threshold={ACCURACY_THRESHOLD} {status}")

            if is_in_ci():
                # 在 GitHub Actions 中把结果写入 step summary 表格
                summary = "### Kimi-K3 Model (MI35x)

"
                summary += "| Model | TP | Accuracy | Threshold | Status |\n"
                summary += "| ----- | -- | -------- | --------- | ------ |\n"
                summary += f"| {KIMI_K3_MODEL_PATH} | {TP_SIZE} | {acc:.3f} | {ACCURACY_
                THRESHOLD} | {status} |\n"
                write_github_step_summary(summary)

            self.assertGreaterEqual(
                acc, ACCURACY_THRESHOLD,
                f"Kimi-K3 accuracy {acc:.3f} below threshold {ACCURACY_THRESHOLD}",
            )
        finally:
            # 无论成败都确保销毁 server 进程，避免污染后续测试
            kill_process_tree(process.pid)
```

评论区精华

Review 层面没有实质交锋：HaiShaw 直接 APPROVED（body 为空）。真正的设计争论体现在 6 个 commit 的演进消息里，作者在提交说明中给出了清晰的工程论证：

- perf 与 accuracy 的先后关系（commit 4d4d1a8）："Throughput on a checkpoint that spends most of an hour loading is not worth an 8-GPU MI35x slot until the kernels are known to emit correct tokens, and that runner is contended enough that a wasted slot costs real time."——在确认内核输出正确 token 之前，不值得为性能占一个稀缺的 8 卡槽位。
- K2.6 不应被整体退役（commit dfc6954）："K3's native MXFP4 weights need gfx95x and mxfp4 does not register on gfx942 ... dropping those jobs would have left MI30x with no Kimi accuracy signal rather than moving it to a new one."——这是对 " 新模型取代旧模型 " 直觉的纠偏，硬件 ISA 约束使覆盖无法迁移。
- 不 pin 镜像的决定（PR body）："Pinning the abandoned rocm720-mi35x-k3-* image line would make a nightly rot over time."——作者选择信任标准 unpinned 镜像，避免废弃镜像线导致 nightly 逐渐腐烂。
- perf nightly 是否应与 accuracy 一起落地 (design): 将 perf 部分拆出推迟到 #34985 跟踪，本 PR 只落 accuracy，避免在精度未经证实前占用稀缺的 8 卡 MI35x 槽位。

- MI300 的 Kimi-K2.6 nightly 是否应退役 (design): 恢复 nightly-8-gpu-kimi-k26 与 nightly-8-gpu-kimi-k26-rocm720，仅替换 MI35x 槽位。
- ROCm 7.0 lane 是否要改动 (design): 7.0 workflow 原样保留，本 PR 只改 7.2 工作流。
- codespell 单字命中导致全 CI 中止 (bugfix): 改为树内一致拼写 unparsable，而非向 .codespellrc 加入 ignore-words-list。

风险与影响

- 风险：本 PR 无源码主路径改动，但作为硬件精度门禁仍有以下风险：
- 外部资源依赖：测试默认从 HuggingFace 拉取 moonshotai/Kimi-K3，MXFP4 权重约 1.5 TB，下载失败或 HF 波动会直接导致 nightly 失败，且浪费 8 卡 MI35x 槽位（job 超时已调至 300 分钟）。
- AITER 内核耦合：测试依赖 SGLANG_AITER_K3_OPT、AITER_SITUV2_A8W4 等 AITER 行为，AITER 升级或改名会导致测试静默失效或误报；AITER_FLYDSL_FORCE 由 AITER 自身消费，仓库内无法 grep 验证，存在黑盒依赖。
- CI 门禁单点敏感：commit bb70e1f 显示一个 codespell 单词（unparsable）就能让 check-pr-test-health 快速失败全部 21 个 base-b-* job，说明门禁链路对微小问题极度敏感。
- 阈值样本量小：0.92 阈值基于一次 0.956 的 green run（0.2% unparsable），统计样本有限，后续硬件或内核抖动可能造成偶发失败。
- 覆盖盲区：MI30x（gfx942）没有 K3 精度覆盖，这是硬件能力限制而非测试遗漏，但值得在文档中持续标注。
- 影响：影响范围集中在 AMD CI 基础设施，不涉及运行时行为：
- 对 AMD 团队：首次获得 Kimi-K3 在 MI355X 上自动化端到端精度信号，避免依赖手动验证；GSM8K 0.92 阈值与既有 Kimi 精度家族对齐，口径统一。
- 对 nightly 资源：ROCm 7.2 lane 的 MI35x 8 卡槽位从 K2.6 迁移到 K3，单 job 时长从约 74 分钟（精度跑完）拉长到包含约 1.5 TB 权重加载的全流程，占用时间显著增加。
- 对覆盖矩阵：MI30x 的 K2.6 精度覆盖保留，ROCm 7.0 lane 不变，总体覆盖不降反升（MI35x 精度从 K2.6 换成新模型 K3）。
- 后续演进：perf 覆盖拆到 #34985 单独跟踪，未来 K3 性能 nightly 会以本 PR 的精度门禁为前提合入。
- 风险标记：外部权重依赖（约 1.5 TB 下载），长耗时 job（超时 300 分钟），CI 门禁对 lint 单词级敏感，仅 MI35x 覆盖（gfx95x 限制），AITER 内核耦合

关联脉络

- PR #32548 AMD Day-0 tracking issue（Kimi-K3 MI355X 吞吐）：PR body 与测试 docstring 反复引用其非投机部署配方，本测试的服务器参数完全对齐该 recipe。
- PR #34985 Kimi-K3 performance coverage 跟踪：本 PR 将 K3 性能 nightly 拆出推迟到该 issue 单独跟踪，commit 4e61bbc 明确说明。

- PR #33832 NightlyBenchmarkRunner API 变更（推断）: commit 4e61bbc 指出 perf 测试基于 #33832 之前的 API 编写导致 setUpClass 崩溃，是范围收缩的直接原因。
- PR #35168 docs: add NVFP4 quantization option to Kimi-K3 deploy panel: 同属 Kimi-K3 支持线，一个补部署文档与量化选项，一个补 ROCm 精度验证，共同完善 K3 在 AMD/Blackwell 上的发布矩阵。