PR #36142 完整报告

sgl-project/sglang

[AMD][CI] Add MiniMax-M3-MXFP8 MI35x nightly perf benchmark

合并时间：2026-08-26 05:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36142

执行摘要

- 一句话：为 M3 新增 MI35x 夜间性能基准，替换 M2.5 槽位
- 推荐动作：值得关注的是其 CI 门控与资源配置设计：精度与性能共用 job、复用已验证 recipe、perf 步骤容错运行，这种模式适合在资源受限的 nightly 矩阵中扩展覆盖。对于关注 AMD 或 MiniMax-M 系列的工程师可精读测试文件与 workflow 变更；对一般开发者，可作为 CI 基准组织方式的参考。

功能与动机

PR body 指出 MiniMax-M3-MXFP8 在 ROCm 7.2 nightly 的 MI35x 上已有精度覆盖（#30613、#33402），但缺少性能覆盖；M2.5 作为第二个 4-GPU MI35x MiniMax job 占用槽位，而 gfx950 原生支持 M3 的 MXFP8 MoE 权重，因此 M3 更值得该槽位。该 PR 将 M2.5 的槽位让给 M3，将释放的时间用于性能基准，且针对推理模型大部分 token 位于 的特点，选择 decode 加权的 1K/1K 形状。

实现拆解

1. 新增性能基准套件（test/registered/amd/perf/mi35x/test_minimax_m3_perf_mi35x.py）：定义 TestNightlyMiniMaxM3PerformanceMI35x，通过 register_amd_ci 注册到 nightly-perf-4-gpu-mi35x-minimax-m3 套件。setUpClass 中构造 model_config，完整复用精度测试的 TP=4 recipe：--quantization mxfp8、aiter 注意力、fp8_e4m3 KV、关闭 radix cache、block-fp8 线性路径（#32036）与 quick INT4 all-reduce（#32230），另加多线程权重加载和 --cuda-graph-max-bs-decode 64，避免把捕获开销花在扫描外的 batch 上。batch 扫描为 [1, 1, 8, 16, 64]，其中重复的 leading 1 提供一次全长度 decode 作为“丢弃行”，用于规避首次长输出的冷启动。
2. 改造 ROCm 7.2 nightly 工作流（.github/workflows/nightly-test-amd-rocm720.yml）：将原 nightly-4-gpu-mi35x-minimax-m25-rocm720 job 替换为 nightly-4-gpu-mi35x-minimax-m3-rocm720，并新增 Performance Test MI35x ROCm 7.2 (4-GPU MiniMax-M3 MXFP8) 步骤，紧随精度步骤之后，且 continue-on-error: true。精度失败即 job 失败，因此不会在错误构建上测吞吐；同时删除了 M2.5 的 job_select 下拉项和 check-all-jobs 的 needs 引用。
3. 资源与门控设计：精度与性能共享同一个 job、同一个已缓存 checkpoint 和同一个 MI35x runner 槽位，步骤顺序天然形成门控；性能步骤容错运行，吞吐偶发波动不会拖垮精度已通过的 job，这是 GLM-5.2-FP8、GLM-5-MXFP4、Kimi-K3、M2.7 等 job 已有的排列方

式。

4. 配套验证与演进约束：在 rocm720 与 rocm724 两个镜像的 MI35x 硬件上实际运行，四个步骤全部通过，两镜像各指标差异在 2% 内；M2.5 的 4-GPU MI35x 精度 job 保留在 ROCm 7.0 nightly（nightly-test-amd.yml），因此其测试文件继续使用，无运行中覆盖丢失。提交历史显示作者曾推入 MI30x 腿和 gfx942 MoE runner 修复，后主动 revert，将 PR 范围收敛到 MI35x。

关键文件：

- test/registered/amd/perf/mi35x/test_minimax_m3_perf_mi35x.py（模块 性能基准；类别 test；类型 test-coverage；符号 TestNightlyMiniMaxM3PerformanceMI35x, setUpClass, test_bench_minimax_m3）：PR 的核心产物：新增 MI35x nightly 性能基准套件，复用精度测试的 TP=4 recipe，避免吞吐回归被配置差异混淆。
- .github/workflows/nightly-test-amd-rocm720.yml（模块 工作流；类别 infra；类型 infrastructure）：将 M2.5 的 4-GPU MI35x job 替换为 M3 精度 + 性能 job，新增 perf 步骤并更新下拉与依赖引用，是资源重分配的执行点。

关键符号：TestNightlyMiniMaxM3PerformanceMI35x, setUpClass, test_bench_minimax_m3

关键源码片段

test/registered/amd/perf/mi35x/test_minimax_m3_perf_mi35x.py

PR 的核心产物：新增 MI35x nightly 性能基准套件，复用精度测试的 TP=4 recipe，避免吞吐回归被配置差异混淆。

```
"""MI35x nightly performance benchmark for MiniMax-M3-MXFP8 (4-GPU, TP=4)."""

import os
import unittest

from sglang.test.ci.ci_register import register_amd_ci
from sglang.test.nightly_bench_utils import generate_simple_markdown_report
from sglang.test.nightly_utils import NightlyBenchmarkRunner
from sglang.test.test_utils import DEFAULT_URL_FOR_TEST, _parse_int_list_env

# 注册为 nightly-perf-4-gpu-mi35x-minimax-m3 套件，预计耗时 5400 秒
register_amd_ci(
    est_time=5400,
    suite="nightly-perf-4-gpu-mi35x-minimax-m3",
    nightly=True,
)

MINIMAX_M3_MODEL_PATH = os.environ.get(
    "MINIMAX_M3_MODEL_PATH", "MiniMaxAI/MiniMax-M3-MXFP8"
)
RESULT_DIR = "performance_results_minimax_m3_mi35x"
MAX_BATCH_SIZE = 64
```

```
class TestNightlyMiniMaxM3PerformanceMI35x(unittest.TestCase):
    """MiniMax-M3-MXFP8 TP=4 在 AMD MI35x 上的吞吐测试。"""

    @classmethod
    def setUpClass(cls):
        cls.base_url = DEFAULT_URL_FOR_TEST
        # 重复的 leading 1 提供一条全长度 decode 作为"丢弃行",
        # 报告 helper 会将其滤除，避免首次长输出污染计时。
        cls.batch_sizes = [1, 1, 8, 16, MAX_BATCH_SIZE]
        cls.input_lens = tuple(_parse_int_list_env("NIGHTLY_INPUT_LENS", "1024"))
        cls.output_lens = tuple(_parse_int_list_env("NIGHTLY_OUTPUT_LENS", "1024"))

        cls.model_config = {
            "name": "TP4+MXFP8+aiterAttn+fp8KV+blockFP8+quickAR",
            "model_path": MINIMAX_M3_MODEL_PATH,
            # 与精度测试 test_minimax_m3_tp4_eval_mi35x.py 的 recipe 保持一致,
            # 额外开启多线程加载，并将 decode graph 上限设为扫描的最大 batch。
            "other_args": [
                "--quantization", "mxfp8",
                "--dtype", "bfloat16",
                "--trust-remote-code",
                "--tp", "4",
                "--attention-backend", "aiter",
                "--kv-cache-dtype", "fp8_e4m3",
                "--disable-radix-cache",
                "--chunked-prefill-size", "8192",
                "--mem-fraction-static", "0.80",
                "--cuda-graph-max-bs-decode", str(MAX_BATCH_SIZE),
                "--max-running-requests", str(MAX_BATCH_SIZE),
                "--model-loader-extra-config",
                '{"enable_multithread_load": true}',
                "--watchdog-timeout", "1200",
            ],
            # block-fp8 线性路径与 quick INT4 all-reduce 在 gfx950 上均为可选,
            # 这些环境变量保证 perf 与精度测试走同一执行路径。
            "env_vars": {
                "SGLANG_USE_AITER": "1",
                "SGLANG_OPT_USE_BF16_ROUTER_GEMM": "0",
                "SGLANG_FORCE_MXFP8_BLOCK_CONVERT": "1",
                "SGLANG_M3_ALLOW_CUSTOM_AR": "1",
                "ROCM_QUICK_REDUCE_QUANTIZATION": "INT4",
                "ROCM_QUICK_REDUCE_CAST_BF16_TO_FP16": "1",
            },
        }

        cls.runner = NightlyBenchmarkRunner(RESULT_DIR, cls.__name__, cls.base_url)
        cls.runner.setup_result_directory()
        cls.runner.full_report = f"## {cls.__name__}\n"
```

```
def test_bench_minimax_m3(self):
    """运行 MiniMax-M3-MXFP8 的 batch 扫描。"""
    old_env = {}
    # ...（循环扫描 batch_sizes，调用 bench_one_batch_server 并汇总报告）
```

评论区精华

该 PR 没有留下 review 评论，HaiShaw 直接批准。有价值的讨论体现在提交历史的演进决策中：一是作者最初在分支中加入了 MI30x nightly job 和 gfx942 MXFP8 MoE runner 修复，随后以 Revert the MI300X leg and the gfx942 MoE fix; keep the PR to MI35x 回退，明确将 PR 聚焦到 MI35x 槽位替换；二是最后一个提交针对 warmup 注释做了修正，澄清 bench_one_batch_server 本身会预热每个 batch，重复的 leading 1 实际提供的是被报告 helper 丢弃的全长度 decode，而非通常理解的 warmup。

- PR 范围收敛：回退 MI30x 腿与 gfx942 修复 (design): PR 聚焦到 MI35x 槽位替换与性能基准，MI30x 相关改动另行处理。
- warmup 注释语义修正 (documentation): 注释已更正，避免后续维护者误解。

风险与影响

- 风险：主要风险集中在 CI 资源重分配和基准可复现性：一是 ROCm 7.2 下 M2.5 的 4-GPU MI35x job 被移除，虽然 M2.5 精度仍在 ROCm 7.0 覆盖，但 M2.5 在 rocm720/rocm724 上的夜间回归能力被有意让出；二是 perf 测试依赖 #32036（block-fp8 线性路径）和 #32230（quick INT4 all-reduce）这类可选路径及一组环境变量（如 SGLANG_FORCE_MXFP8_BLOCK_CONVERT、ROCM_QUICK_REDUCE_QUANTIZATION），这些路径后续若被改动或默认值变化，可能使基准结果漂移或报错；三是测试使用 bench_one_batch_server 且 TP=4，与 cookbook 发布的 tp8 bench_serving 行不具备直接可比性，文档注释已说明应视为形状参考；四是 continue-on-error: true 意味着 perf 步骤失败不会暴露 job 失败，需要人工关注日志。
- 影响：影响限于 AMD nightly CI：M3 的 MI35x 性能从此有持续回归监控，M2.5 在 ROCm 7.2 的 4-GPU 槽位让给 M3，净 runner 成本基本不变；对用户无直接可见影响，但有助于在发布前捕获 M3 在 gfx950 上的吞吐回归。团队侧需要留意 perf 步骤的失败日志由于 continue-on-error 不会红 job，且 M2.5 的 ROCm 7.2 覆盖缺失需依赖 7.0。
- 风险标记：CI 资源重分配，M2.5 的 ROCm 7.2 覆盖移除，基准依赖可选路径，continue-on-error 可能掩盖 perf 失败，数值与 cookbook 形状不可直接对比

关联脉络

- PR #36097 Fix MXFP8 MoE weight sizing for non-gated models: 同为 MXFP8 MoE 相关修复，可能影响 M3 在非 gated 场景下的权重尺寸与加载路径。
- PR #36290 [AMD][CI] Adjust MI300 score API performance thresholds: 同为 AMD CI 性能维护，体现 AMD nightly 性能门槛的持续调优脉络。
- PR #33021 [AMD] Drop redundant FP8 bpreshuffle scale transpose via fused AR kernel: AMD 侧 FP8/MoE 性能优化，与 M3 的 block-fp8 线性路径可能共享 kernel 基础。