PR #21773 完整报告

sgl-project/sglang

[AMD][CI] Add GLM-5-MXFP4 accuracy and perf nightly tests for MI35x

合并时间：2026-04-15 09:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/21773

执行摘要

- 一句话：为 AMD MI35x 添加 GLM-5-MXFP4 模型的夜间准确性及性能测试，并清理过时任务。
- 推荐动作：建议工程团队了解 AMD CI 测试的配置模式，但 PR 本身变更较为直白，无需深入设计决策。重点关注测试配置参数（如准确性阈值、性能输入长度）是否符合项目标准。

功能与动机

根据 PR body，动机是添加 AMD GLM-5-MXFP4 模型的 nightly 测试以监控其准确性和性能，并移除过时的 GLM-5（BF16 NSA）CI 作业，因为这些已被 GLM-5.1 和 GLM-5-MXFP4 取代。

实现拆解

1. 新增准确性测试文件：在 test/registered/amd/accuracy/mi35x/test_glm5_mxfp4_eval_mi35x.py 中，定义 ModelConfig 类和相关函数，配置模型路径、TP 大小、准确性阈值（0.90）及服务器启动参数（如禁用 radix cache、设置上下文长度等）。关键函数 get_glm5_mxfp4_models 返回测试配置，run_gsm8k_benchmark 执行评估。
2. 新增性能测试文件：在 test/registered/amd/perf/mi35x/test_glm5_mxfp4_perf_mi35x.py 中，创建 TestGLM5MXFP4PerfMI35x 类，设置基准测试参数（批量大小、输入 / 输出长度），并生成简化 Markdown 报告。函数 generate_simple_markdown_report 负责报告生成，避免零除错误。
3. 更新 CI 工作流：修改 .github/workflows/nightly-test-amd.yml 和 nightly-test-amd-rocm720.yml，添加新作业 nightly-8-gpu-mi35x-glm5-mxfp4（及 ROCm 7.2 变体），并移除旧作业 nightly-8-gpu-glm5 和 nightly-8-gpu-glm51-mxfp4。
4. 测试验证：PR 中提供了测试运行链接，确保 CI 通过，并遵循现有 AMD 测试模式。

关键文件：

- test/registered/amd/accuracy/mi35x/test_glm5_mxfp4_eval_mi35x.py（模块 AMD 准确性测试；类别 test；类型 test-coverage；符号 get_model_path, ModelConfig, post_init, get_display_name）：新增 GLM-5-MXFP4 模型的准确性测试文件，用于评估其在 GSM8K 数据集上的表现，设置准确性阈值并配置服务器参数。
- test/registered/amd/perf/mi35x/test_glm5_mxfp4_perf_mi35x.py（模块 AMD 性能测试；类别 test；类型 test-coverage；符号 generate_simple_markdown_report,

get_model_path, TestGLM5MXFP4PerfMI35x, setUpClass）：新增性能测试文件，用于基准测试 GLM-5-MXFP4 模型在 MI35x 上的吞吐量和延迟。

- .github/workflows/nightly-test-amd.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：更新 CI 工作流文件，添加 GLM-5-MXFP4 夜间测试作业并移除过时的 GLM-5 作业，确保测试自动化。
- .github/workflows/nightly-test-amd-rocm720.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：更新 ROCm 7.2 版本的 CI 工作流文件，同步添加和移除测试作业，确保跨 ROCm 版本的一致性。

关键符号：get_model_path, ModelConfig.post_init, get_glm5_mxfp4_models, generate_simple_markdown_report, TestGLM5MXFP4PerfMI35x.setUpClass, test_bench_one_batch

关键源码片段

test/registered/amd/accuracy/mi35x/test_glm5_mxfp4_eval_mi35x.py

新增 GLM-5-MXFP4 模型的准确性测试文件，用于评估其在 GSM8K 数据集上的表现，设置准确性阈值并配置服务器参数。

```
def get_glm5_mxfp4_models() -> List[ModelConfig]:
    """Get GLM-5-MXFP4 model configurations for MI35x."""
    model_path = get_model_path() # 获取模型路径，优先环境变量，其次本地路径，最后 HuggingFace ID
    return [
        ModelConfig(
            model_path=model_path,
            tp_size=8, # 使用 8 GPU 张量并行
            accuracy_threshold=0.90, # 设置准确性阈值为 90%
            timeout=5400, # 测试超时时间（秒）
            variant="mxfp4", # 标识量化变体
            other_args=[
                "--trust-remote-code", # 信任远程代码以加载模型
                "--chunked-prefill-size", "131072", # 设置分块预填充大小
                "--disable-radix-cache", # 禁用 radix 缓存
                "--mem-fraction-static", "0.85", # 静态内存分配比例
                "--context-length", "4096", # 上下文长度
                "--model-loader-extra-config", '{"enable_multithread_load": true}', # 启用多线程加载
                "--watchdog-timeout", "1200", # 看门狗超时
            ],
            env_vars={"SGLANG_USE_AITER": "1"}, # 设置环境变量启用异步迭代器
        ),
    ]
```

test/registered/amd/perf/mi35x/test_glm5_mxfp4_perf_mi35x.py

新增性能测试文件，用于基准测试 GLM-5-MXFP4 模型在 MI35x 上的吞吐量和延迟。

```
def generate_simple_markdown_report(results: List[BenchmarkResult]) -> str:
    """Generate a simplified markdown report without traces and cost columns.
```

```
    Skips the first result if it's a warmup run (duplicate batch_size).
    """
    model_header = results[0].model_path
    if results[0].run_name and results[0].run_name != "default":
        model_header += f" ({results[0].run_name})"

    gpu_config = os.getenv("GPU_CONFIG", "MI35x") # 获取 GPU 配置，默认 MI35x
    if gpu_config:
        model_header += f" [{gpu_config}]"

    summary = f"### {model_header}\n"
    summary += "| batch size | input len | latency (s) | input throughput (tok/s) | output
    throughput (tok/s) | ITL (ms) |\n"
    summary += "| ---------- | --------- | ----------- | ------------------------ | ------------------------ | --------
    |\n"

    # 跳过预热运行的结果
    report_results = (
        results[1:]
        if len(results) > 1 and results[0].batch_size == results[1].batch_size
        else results
    )

    for result in report_results:
        # 计算 Inter-Token Latency (ITL)，避免除零错误
        itl = (
            1 / (result.output_throughput / result.batch_size) * 1000
            if result.output_throughput > 0
            else 0
        )
        summary += f"| {result.batch_size} | {result.input_len} | {result.latency:.2f} | {result.input_
        throughput:.2f} | {result.output_throughput:.2f} | {itl:.2f} |\n"

    return summary
```

评论区精华

review 中主要讨论了环境变量设置的位置（应移至 setUpClass 以避免副作用）、数值解析的安全性（建议使用 int() 替代 ast.literal_eval）、零除错误防护、以及添加推理解析器参数。作者回应遵循现有 AMD 测试模式，未对评论做修改，因此部分疑虑未解决。

- 环境变量设置位置 (design): 作者回应遵循现有模式，未做改动。
- 数值解析安全性 (correctness): 未在 PR 中修改，可能后续处理。
- 零除错误防护 (correctness): PR 中已实现检查（if result.output_throughput > 0），但评论指出应更明确。
- 添加推理解析器参数 (design): 作者添加了这些参数到性能测试配置中。

风险与影响

- 风险：技术风险包括：测试文件中的模块级环境变量设置可能导致进程间副作用（如 gemini-code-assist 指出）；数值解析使用 ast.literal_eval 可能因前导零引发 SyntaxError；CI 配置变更若错误可能影响夜间测试的运行稳定性。这些风险局限于测试层面，不影响生产代码。
- 影响：对用户无直接影响，因为这是内部测试扩展。对系统而言，增强了 AMD 平台 GLM-5-MXFP4 模型的自动化监控能力，有助于早期发现性能或准确性回归。对团队，提供了标准化的测试模板，便于未来类似模型的集成。
- 风险标记：环境变量副作用，数值解析风险，CI 配置变更

关联脉络

- PR #22768 ci: skip approval for nightly gb200 runs, keep for manual triggers: 同样涉及 CI 工作流调整，共享类似的权限和触发逻辑。
- PR #22537 Add runai-model-streamer into Python packages installed in Dockerfile and fix NotADirectoryError Docker regression: 基础设施相关，涉及依赖和 Docker 配置，与本 PR 的 CI 测试扩展有协同作用。