PR #27242 完整报告

sgl-project/sglang

[MUSA][23/N] CI: Fix torchada preflight lock cleanup and add LLM server smoke test

合并时间：2026-06-08 23:45

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27242

执行摘要

- 一句话：MUSA CI: 修复 torchada 锁清理并新增 LLM Server 冒烟测试
- 推荐动作：建议精读并关注设计决策：CI 注册模式（AST 解析注册）、锁清理的防御性编程模式（进程检测 + 目录检查）值得在类似场景复用。对于 MUSA 平台维护者，应关注测试稳定性；对于 CI 团队，可参考其工作流定义模板。

功能与动机

PR 旨在提升 MUSA CI 的可靠性，解决 sgl-kernel 发布流水线中 observed 的构建挂起问题（原因为 torchada 扩展锁文件残留），并为 MUSA 后端增加 LLM server 启动和基本生成请求的测试覆盖，确保 nightly CI 中 MUSA 平台的基本可用性。

实现拆解

1. 注册 MUSA CI 后端：在 python/sglang/test/ci/ci_register.py 中添加 register_musa_ci 标记函数和 HWBackend.MUSA 枚举，使测试能通过 AST 解析注册。同时将 register_musa_ci 加入 __all__ 和 REGISTER_MAPPING。
2. 新增 MUSA LLM Server 冒烟测试：创建 test/registered/musa/test_llm_server_smoke_musa.py，定义 TestMusaDeepSeekV2LiteChatServerSmoke 类，继承 DefaultServerBase，配置 DeepSeek-V2-Lite-Chat 模型和特定参数（fa3 注意力、禁用分段 CUDA Graph 等）。包含四个测试用例：test_health（健康检查）、test_health_generate、test_send_receive_chat_message_contains_beijing（验证聊天补全会话接口）、test_generate（验证生成接口非空）。使用 @unittest.skipIf 确保只在具备 MUSA 的设备上运行，并用 register_musa_ci 注册为夜间 1-GPU 套件，预计耗时 1200 秒。
3. 更新测试套件运行配置：在 test/run_suite.py 中，将 "musa" 映射到 HWBackend.MUSA，在 PER_COMMIT_SUITES 中添加 HWBackend.MUSA: []，在 NIGHTLY_SUITES 中添加 HWBackend.MUSA: ["nightly-musa-1-gpu"]，并将 HWBackend.MUSA 加入 _SUITE_CHECKED_BACKENDS，以启用套件验证。
4. 配置 MUSA CI 工作流：在 .github/workflows/nightly-test-musa.yml 中添加新的 job nightly-test-llm-server-1-gpu-musa，指定 runner、超时 240 分钟，执行依赖安装后通过 run_suite.py 运行 nightly-musa-1-gpu 套件。同时将多个 job 的安装依赖超时从 10 分钟提升至 15 分钟，以应对安装负载。

5. 修复 torchada 扩展锁清理：在 scripts/ci/musa/musa_install_dependency.sh 中，在升级 torchada 之前，先检测是否有活跃的构建进程（mcc、ninja、torchada_cpp_ops）；若存在则拒绝清理锁文件并退出（避免竞态），否则删除 ${HOME}/.cache/torch_extensions 下的锁文件。在 .github/workflows/release-whl-kernel.yml 中也加入类似步骤，确保发布流水线同样受益。

关键文件：

- test/registered/musa/test_llm_server_smoke_musa.py（模块 冒烟测试；类别 test；类型 test-coverage；符号 TestMusaDeepSeekV2LiteChatServerSmoke, test_health, test_health_generate, test_send_receive_chat_message_contains_beijing）：新增 MUSA LLM Server 冒烟测试文件，包含四个测试用例覆盖健康检查和生成端点，是本次变更的核心测试配套。
- python/sglang/test/ci/ci_register.py（模块 CI 注册；类别 test；类型 test-coverage；符号 register_musa_ci）：添加 MUSA CI 注册函数和枚举，使测试能被 AST 解析并分配到 MUSA 后端运行。
- .github/workflows/nightly-test-musa.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：新增 LLM server 冒烟测试 job 并调整超时时间，是 MUSA CI 工作流的核心变更。
- test/run_suite.py（模块 套件配置；类别 test；类型 test-coverage）：更新套件运行配置以支持 MUSA 后端，包括 HW_MAPPING、PER_COMMIT_SUITES、NIGHTLY_SUITES 和 _SUITE_CHECKED_BACKENDS。
- scripts/ci/musa/musa_install_dependency.sh（模块 安装脚本；类别 infra；类型 infrastructure）：实现 torchada 锁清理核心逻辑：进程检测和锁文件删除，是解决构建挂起的关键修复。
- .github/workflows/release-whl-kernel.yml（模块 发布流水线；类别 infra；类型 infrastructure）：对 kernel 发布流水线也应用 torchada 锁清理，确保发布过程不因锁文件残留挂起。
- .github/workflows/pr-test-musa.yml（模块 PR 测试；类别 infra；类型 infrastructure）：调整安装依赖超时时间，与其他 MUSA 工作流保持一致。

关键符号：register_musa_ci, TestMusaDeepSeekV2LiteChatServerSmoke, test_health, test_health_generate, test_send_receive_chat_message_contains_beijing, test_generate

关键源码片段

test/registered/musa/test_llm_server_smoke_musa.py

新增 MUSA LLM Server 冒烟测试文件，包含四个测试用例覆盖健康检查和生成端点，是本次变更的核心测试配套。

```
import os
import unittest

import requests
import torch
```

```
from sglang.test.ci.ci_register import register_musa_ci
from sglang.test.server_fixtures.default_fixture import DefaultServerBase

# 注册为 MUSA nightly 1-GPU 套件，预计耗时 1200 秒
register_musa_ci(est_time=1200, suite="nightly-musa-1-gpu", nightly=True)

_REQUEST_TIMEOUT = 60


@unittest.skipIf(
    not (hasattr(torch, "musa") and torch.musa.is_available()),
    "MUSA device not available",
)
class TestMusaDeepSeekV2LiteChatServerSmoke(DefaultServerBase):
    """MUSA LLM server 冒烟测试：启动、健康检查、非空生成。"""

    model = os.getenv("SGLANG_MUSA_LLM_MODEL", "deepseek-ai/DeepSeek-V2-Lite-Chat")
    served_model_name = "deepseek-v2-lite-chat"
    other_args = [
        "--trust-remote-code",
        "--served-model-name", served_model_name,
        "--attention-backend", "fa3", # 使用 flash attention 3 后端
        "--cuda-graph-max-bs", "32",
        "--tp-size", "1", # 单 GPU 测试
        "--chunked-prefill-size", "-1",
        "--disable-piecewise-cuda-graph", # 禁用分段 CUDA Graph 简化测试
        "--context-length", "4096",
        "--max-total-tokens", "8192",
        "--max-running-requests", "4",
    ]

    def test_health(self):
        """验证 /health 返回 200。"""
        resp = requests.get(self.base_url + "/health", timeout=10)
        self.assertEqual(resp.status_code, 200, resp.text)

    def test_health_generate(self):
        """验证 /health_generate 返回 200。"""
        resp = requests.get(self.base_url + "/health_generate", timeout=_REQUEST_TIMEOUT)
        self.assertEqual(resp.status_code, 200, resp.text)

    def test_send_receive_chat_message_contains_beijing(self):
        """验证聊天补全会话接口能够返回包含 'Beijing' 的回答。"""
        resp = requests.post(
            self.base_url + "/v1/chat/completions",
            json={
                "model": self.served_model_name,
                "messages": [{"role": "user", "content": "What is the capital of China? Answer in one
                word."}],
```

```
                "temperature": 0.0,
                "max_tokens": 16,
            },
            timeout=_REQUEST_TIMEOUT,
        )
        self.assertEqual(resp.status_code, 200, resp.text)
        body = resp.json()
        self.assertIn("choices", body)
        content = body["choices"][0]["message"]["content"]
        # 注意：大小写不敏感断言，避免因模型输出大小写变化导致假失败
        self.assertIn("beijing", content.lower())

    def test_generate(self):
        """验证 /generate 端点返回非空文本。"""
        resp = requests.post(
            self.base_url + "/generate",
            json={
                "text": "The capital of France is",
                "sampling_params": {"temperature": 0.0, "max_new_tokens": 16},
                "stream": False,
            },
            timeout=_REQUEST_TIMEOUT,
        )
        self.assertEqual(resp.status_code, 200, resp.text)
        body = resp.json()
        if isinstance(body, list):
            body = body[0]
        self.assertIn("text", body)
        self.assertGreater(len(body["text"].strip()), 0)


if __name__ == "__main__":
    unittest.main()
```

python/sglang/test/ci/ci_register.py

添加 MUSA CI 注册函数和枚举，使测试能被 AST 解析并分配到 MUSA 后端运行。

```
# 在 HWBackend 枚举中添加 MUSA 后端
class HWBackend(Enum):
    CPU = auto()
    CUDA = auto()
    AMD = auto()
    NPU = auto()
    XPU = auto()
    MUSA = auto() # 新增 MUSA 后端枚举

# 注册函数：作为 AST 标记使用，运行时无操作
def register_musa_ci(
    est_time: float,
```

```
    suite: Optional[str] = None,
    nightly: bool = False,
    disabled: Optional[str] = None,
    *,
    stage: Optional[str] = None,
    runner_config: Optional[str] = None,
):
    """Marker for MUSA CI registration（parsed via AST; runtime no-op）。"""
    return None

# 在 REGISTER_MAPPING 中添加映射
REGISTER_MAPPING = {
    "register_cpu_ci": HWBackend.CPU,
    "register_cuda_ci": HWBackend.CUDA,
    "register_amd_ci": HWBackend.AMD,
    "register_musa_ci": HWBackend.MUSA, # 新增映射
    "register_npu_ci": HWBackend.NPU,
    "register_xpu_ci": HWBackend.XPU,
}
```

评论区精华

> 进程检测方法：gemini-code-assist[bot] 建议使用 pgrep 替代 pslgrep 以避免脚本自身进程或目录名称包含关键字导致的误判。最终代码采纳了 pgrep -af ... | awk -v self="$$" '$1 != self' 模式，提升了检测鲁棒性。
>
> 目录存在性检查：同样来自 gemini-code-assist[bot] 的建议：在删除锁文件前应检查目录是否存在，避免 find 输出 stderr 错误。最终代码添加了 if [-d "$torch_extensions_dir"] ; then ... fi 包裹。
>
> 断言大小写不敏感：gemini-code-assist[bot] 指出 self.assertIn("Beijing", content) 是大小写敏感的，建议使用 content.lower() 避免因模型输出大小写变化导致假失败。最终代码改为 self.assertIn("beijing", content.lower())。
>
> 超时时间调整：yeahdongcn 要求将多个 job 的安装依赖超时统一从 10 分钟改为 15 分钟，以避免超时。该请求被采纳并在工作流中批量应用。

- 进程检测使用 pgrep 替代 pslgrep (correctness): 最终采纳 pgrep -af ... | awk -v self="$$" '$1 != self'，提升了检测鲁棒性。
- 锁文件删除前的目录存在性检查 (correctness): 添加 if [-d "$torch_extensions_dir"]; then ... fi 包裹。
- 断言大小写不敏感 (correctness): 改为 self.assertIn("beijing", content.lower())。
- 统一安装依赖超时时间至 15 分钟 (infra): 在 nightly-test-musa.yml 和 pr-test-musa.yml 中批量更新。

风险与影响

- 风险：

- 锁清理逻辑误判风险：如果系统中存在与 mcc、ninja 或 torchada_cpp_ops 同名的非构建进程（例如代码编辑器、CI 子进程），可能导致误判而阻止清理。当前使用 pgrep 并排除自身 PID 来缓解，但仍有可能覆盖不完整。建议后续持续关注。
- 冒烟测试可靠性：新增测试基于 DefaultServerBase，依赖预缓存模型和稳定环境。模型加载失败、网络波动或 MUSA 驱动问题可能导致 flakes。测试超时设置为 120 分钟足够宽松，但仍可能因环境问题不稳定。
- CI 作业时长增加：新增 LLM server 测试预计耗时 1200 秒（20 分钟），加上其他 job 的超时提升，整体 MUSA nightly CI 运行时间有所增加，可能影响资源调度。
- 多 GPU 场景覆盖缺失：当前仅 1-GPU 测试，未涉及 TP/EP 等并行策略，生产风险暴露不全面。

- 影响：
  - 对用户：MUSA 硬件用户将受益于更可靠的 CI，确保 nightly 构建中的 SGLang 服务器在 MUSA 上可用。无直接用户功能变更。
  - 对系统：新增 nightly CI 作业增加约 20 分钟运行时长，但属预期内。
  - 对团队：需要维护新测试和锁清理脚本；测试中硬编码了 DeepSeek-V2-Lite-Chat 参数，未来模型更新需同步。
  - 风险标记：锁清理逻辑误判风险，冒烟测试依赖环境稳定性，CI 作业时长增加

关联脉络

- PR #27537 [MUSA] bump torchada version to 0.1.59 and workaround PCG limitation.: 同样涉及 MUSA 后端 torchada 依赖管理，该 PR 升级 torchada 版本，本 PR 修复锁清理问题，为版本升级提供更稳定的 CI 环境。