PR #32984 完整报告

sgl-project/sglang

[MLX] Upgrade to Torch 2.13/MLX 0.32+ and redesign the Torch-MLX tensor bridge

合并时间：2026-08-22 09:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32984

执行摘要

- 一句话：升级 Torch 2.13/MLX 0.32，重设计零拷贝张量桥
- 推荐动作：值得精读，尤其是 python/sglang/srt/utils/tensor_bridge.py 的所有权契约（owned vs borrowed 三个层次）与负 stride 物化设计：它把「双框架共用同一块 Metal 内存」的竞态、生命周期问题显式化，并用 RLock 串行化完整穿越、用 DLPack 消除 payload 拷贝。对任何做 Torch + 其他运行时张量桥的团队都有直接借鉴意义；runtime.py 的版本门控矩阵（拒 rc/dev、Torch 锁系列、MLX 只设下限）也是可复用的 fail-fast 模式。

功能与动机

PR body 给出三层动机：其一，MLX 0.32 可以直接导入 Torch MPS 分配，而 Torch 2.13 改变了运行时与 DLPack 边界语义，旧桥不再适配；其二，旧桥把 MPS 张量先搬到 CPU 再经 NumPy 中转，双向都付出 payload 大小的拷贝，且 borrowed-vs-owned 生命周期被隐式处理，参数替换或 GC 后容易引用失效存储；其三，diffusion 的 SGLANG_USE_MLX norm override 在生产图中实际不可达——CustomOp.dispatch_forward() 对 MPS 路由到 forward_native，Sana 仪器化确认每 DiT forward 的 58 个 norm 零调用，强制全部穿过 Torch-MLX 桥反而使 denoise step 中位数 +62.5%。因此「删除不可达路径、收紧版本门控」是数据驱动的决策。

实现拆解

1. 依赖升级与运行时提前门控：python/pyproject_other.toml 的 srt_mps extra 将 torch==2.11.0 提升到 torch==2.13.0、mlx 改为 mlx>=0.32.0，并新增 torchcodec==0.15.0、torchvision==0.28.0；版本继续钉在 srt_mps 内，all_mps 继承，CPU 与独立 diffusion_mps 依赖列表不变。新增 python/sglang/srt/hardware_backend/mlx/runtime.py，_validate_runtime() 用 packaging.version 同时校验 stable Torch 2.13.x、stable MLX>=0.32.0、torch.backends.mps.is_available() 与 mlx.metal.is_available()，失败信息统一带「reinstall with the srt_mps extra」的可操作指引；server_args.py 在 _run_resolution_pipeline() 中新增 _handle_hardware_runtime_validation()，位于模型路径解析、权重下载与 dummy-model 短路之前；use_mlx() 使用 lru_cache，未设置 SGLANG_USE_MLX 时不导入 mlx（由子进程测试验证 sys.modules）。

2. 张量桥所有权模型重构：tensor_bridge.py 删除全局 _MLX_AVAILABLE 探测与 NumPy/CPU 中转，改用 _mlx_core() 懒加载与 _BRIDGE_LOCK（RLock）经 _serialized_bridge 装饰器串行化所有穿越。torch_to_mlx() 语义改为「独立 MLX 拷贝」：MPS 输入 mx.asarray(copy=True) 后立即 mx.eval 物化；CPU 输入永远构造 MLX-owned 存储，float64 走 mx.cpu stream、complex128 显式报错。新增 MlxTensorView（同时持有 detached Torch tensor 与 MLX array 两个强引用，含 matches() 供参数替换后校验）与 borrow_torch_tensors()（批量借用、单次 fence）。mlx_call() / mlx_call_multi() 先校验目标设备（cuda 等在开跑前拒绝），一次 torch.mps.synchronize() 后零拷贝导入，所有输出共享一次 mx.eval 边界，经 DLPack 导出（CPU 目标用 dl_device=(1,0) 显式视图）；_has_negative_stride() 预检负 stride 输出并物化，因为 Torch 2.13 的 DLPack 导入器遇负 stride 会 abort 进程而非抛 Python 异常。
3. diffusion MPS 路径收敛：删除 python/sglang/kernels/ops/diffusion/common/fallback_mps.py（122 行 MLX 加速实现）；fallback_torch.py 的 norm_infer_native() 改用 F.rms_norm / F.layer_norm，在 fp32 下累加 bias 后一次性舍入回输入 dtype，保留 optional-bias 与「fp16/bf16 输入 + fp32 参数」契约；rotary、scale/shift 原本就是 Torch 别名直接清理。移除 diffusion 侧 SGLANG_USE_MLX 运行时处理，文档明确该变量只作用于 SRT 服务。
4. Torch 2.13 兼容桩与配套迁移：_platform_stubs.py 新增 _KernelInterface、_OutOfResources、_PTXASError、_CompiledKernel、_ASTSource、_GPUTarget 类符号以及 triton.compiler.compiler.triton_key，避免 catch-all meta-path finder 把类名物化为模块，保证 MPS 上 Inductor import 成功。rebase 后把 VAE/Transformer loader 迁到 upstream 的 unified residency API；scheduler.py、one_batch.py、overrides.py、profiler.py、kv_cache_builder.py 等仅调整 use_mlx 导入来源。
5. 测试与 CI 配套：新增 4 个注册测试——test_tensor_bridge.py（603 行：owned/borrowed/persistent 生命周期、源变异与删除、单次 fence、单次 eval、负 stride abort-safe 子进程、CPU float64、无效目标 fail-fast、autograd 分离）；test_runtime.py（版本门控矩阵、缺失 MLX/MPS/Metal 的可操作报错、dummy 模型前中止）；test_diffusion_torch_fallback.py（同时注册 MLX 与 CPU/ARM64 套件，锁定 fp16/bf16+fp32参数精度）；test_diffusion_import_isolation.py（未启用时不导入mlx）。4 个文件全部注册进 stage-a-unit-test-mlx，diffusion fallback 测试额外注册进 base-a-test-cpu、base-b-test-cpu、base-b-test-cpu-arm64。

关键文件：

- python/sglang/srt/utils/tensor_bridge.py（模块 张量桥；类别 source；类型 core-logic；符号 _serialized_bridge, MlxTensorView, borrow_torch_tensors, torch_to_mlx）：桥核心重设计：从 CPU/NumPy 中转改为 DLPack 零拷贝，显式区分 owned/borrowed 所有权，新增 _serialized_bridge 锁、MlxTensorView、borrow_torch_tensors、mlx_call / mlx_call_multi，并处理 CPU float64、complex128、负 stride 等边界。
- python/sglang/srt/hardware_backend/mlx/runtime.py（模块 运行时门控；类别 source；类型 runtime-gate；符号 _is_stable_series, _is_stable_at_least, _validate_runtime, use_mlx）：新增运行时门控：集中校验 stable Torch 2.13.x、MLX>=0.32、MPS 与 MLX Metal 可用性，是 fail-fast 策略的执行主体。

- test/registered/unit/utils/test_tensor_bridge.py（模块 桥接测试；类别 test；类型 test-coverage；符号 TestTensorBridgeImport, test_import_does_not_eagerly_import_mlx, TestTensorBridgeCpu, TestTensorBridgeMetalSharing）：603 行桥测试：覆盖 owned/borrowed/persistent 生命周期、源变异与删除、单次 fence、单次 eval、负 stride abort-safe 子进程、CPU float64 保持、无效目标 fail-fast。
- python/sglang/_platform_stubs.py（模块 平台桩；类别 source；类型 core-logic；符号 _KernelInterface, _OutOfResources, _PTXASError, _CompiledKernel）：为 Torch 2.13 补全 Inductor 在 MPS 上 import 所需的 Triton 类符号，避免 catch-all finder 把类名物化为模块。
- test/registered/unit/hardware_backend/mlx/test_runtime.py（模块 门控测试；类别 test；类型 test-coverage；符号 _fake_mlx, TestMlxRuntime, test_version_gates, test_disabled_backend_does_not_import_mlx）：版本门控矩阵测试：拒 2.14/2.12/rc/dev/0.31，接受 0.32+ 及未来 stable minor；验证 disabled 时不导入 mlx、dummy 模型前中止、缺失组件报错可操作。
- python/sglang/kernels/ops/diffusion/common/fallback_torch.py（模块 扩散回退；类别 infra；类型 infrastructure；符号 norm_infer_native）：diffusion MPS fallback 的数值核心：norm_infer_native 改用 F.rms_norm / F.layer_norm，修复 bias 累加时机精度问题，保留 optional-bias 与输入 dtype 契约。
- python/sglang/kernels/ops/diffusion/common/fallback_mps.py（模块 扩散回退；类别 infra；类型 deletion；符号 norm_infer_native, triton_one_pass_rms_norm_native, rms_norm_fn_native）：整个 MLX 加速 diffusion fallback 被删除：该路径在生产 MPS 图中不可达，且强制穿越桥更慢（Sana denoise step +62.5%）。
- python/sglang/srt/server_args.py（模块 启动参数；类别 source；类型 core-logic；符号 _handle_hardware_runtime_validation）：运行时校验的接入点：在解析管线最早阶段调用 use_mlx()，早于模型路径解析、下载与 dummy 短路。

关键符号：mlx_call, mlx_call_multi, torch_to_mlx, mlx_to_torch, borrow_torch_tensors, MlxTensorView, _torch_to_mlx, _serialized_bridge, _validate_runtime, use_mlx, _handle_hardware_runtime_validation, norm_infer_native

关键源码片段

python/sglang/srt/hardware_backend/mlx/runtime.py

新增运行时门控：集中校验 stable Torch 2.13.x、MLX>=0.32、MPS 与 MLX Metal 可用性，是 fail-fast 策略的执行主体。

```
# python/sglang/srt/hardware_backend/mlx/runtime.py（新增，核心逻辑）
# 目标：把版本 / 设备校验前移到 ServerArgs 构造的最早阶段，
# 在模型路径解析、权重下载甚至 dummy-model 短路之前就 fail fast。

from functools import lru_cache

import torch
from packaging.version import InvalidVersion, Version
```

```
from sglang.srt.environ import envs

_MIN_MLX_VERSION = Version("0.32.0")
_SUPPORTED_TORCH_SERIES = (2, 13)


def _is_stable_series(raw_version: object, series: tuple[int, int]) -> bool:
    """只接受 stable 系列：拒绝 rc/dev，也拒绝 2.14 等未验证的大版本。"""
    try:
        version = Version(str(raw_version))
    except InvalidVersion:
        return False
    return not version.is_prerelease and (version.major, version.minor) == series


def _is_stable_at_least(raw_version: object, minimum: Version) -> bool:
    """MLX 只设下限：0.32.0 之后的 stable 小版本（含 0.33）都放行。"""
    try:
        version = Version(str(raw_version))
    except InvalidVersion:
        return False
    return not version.is_prerelease and version >= minimum


@lru_cache(maxsize=1)
def _validate_runtime() -> None:
    try:
        import mlx.core as mx
    except ImportError:
        raise RuntimeError(
            "SGLANG_USE_MLX requires stable Torch 2.13.x and MLX >= 0.32.0, "
            "but MLX is not installed; reinstall with the srt_mps extra"
        ) from None
    mlx_version = getattr(mx, "__version__", None)
    torch_version = getattr(torch, "__version__", None)
    if not _is_stable_series(
        torch_version, _SUPPORTED_TORCH_SERIES
    ) or not _is_stable_at_least(mlx_version, _MIN_MLX_VERSION):
        raise RuntimeError(
            "SGLANG_USE_MLX requires stable Torch 2.13.x and MLX >= 0.32.0; "
            f"found Torch {torch_version or 'unknown'} + MLX {mlx_version or 'unknown'}; "
            "reinstall with the srt_mps extra"
        )

    # 版本达标后还要确认 MPS 设备与 MLX Metal 设备真实可用，
    # 给出可操作报错而不是让后续在随机位置崩溃。
    mps_backend = getattr(torch.backends, "mps", None)
    is_mps_available = getattr(mps_backend, "is_available", None)
    if not callable(is_mps_available) or not is_mps_available():
```

```
            raise RuntimeError("SGLANG_USE_MLX requires an available PyTorch MPS device")

        metal = getattr(mx, "metal", None)
        is_available = getattr(metal, "is_available", None)
        if not callable(is_available) or not is_available():
            raise RuntimeError("SGLANG_USE_MLX requires an available MLX Metal device")


@lru_cache(maxsize=1)
def use_mlx() -> bool:
    """显式启用才触发校验；未设置 SGLANG_USE_MLX 时保持懒导入，不碰 mlx。"""
    enabled = bool(envs.SGLANG_USE_MLX.get())
    if enabled:
        _validate_runtime()
    return enabled
```

评论区精华

评审核心围绕桥的边界语义与数值精度：

- noob-se7en 发现 CPU float64 导出缺陷：「My local test failed here because MLX CPU float64 array is exported here as a PyTorch float64 MPS tensor. (PyTorch MPS does not support float64)」。yeahdongcn 修复为 CPU float64 在 mx.cpu stream 构造，并在 _prepare_mlx_export() 对 MPS 目标显式拒绝 float64，配套新增 test_mlx_call_multi_preserves_cpu_float64。
- alexnails 质疑：「there can be race condition / deadlock on MLX vs Torch borrow on exit?」yeahdongcn 复现并修复，补充了借用期存活与负 stride 物化的处理。
- alexnails 对 fallback_torch.py 做精度审计：bias 被加到已窄化回输入 dtype 的结果上，旧代码是先 fp32 累加再一次性舍入；实测 bf16+fp32 参数时 max err 从 1.5e-2 升到 2.7e-2。yeahdongcn 修复后由 test_norm_infer_preserves_input_dtype_with_fp32_parameters 锁定。
- noob-se7en 指出 CI 依赖集冲突：「The MLX Stage A workflow installs srt_mps,test, but this test imports diffusion modules requiring diffusers, causing a clean-install failure.」扩散导入冒烟被拆到 test_diffusion_import_isolation.py，Stage A 依赖集保持不变。
- alexnails 质疑新增依赖：「we are adding torchcodec?」yeahdongcn 解释其用于多模态音视频解码，与 cuda、xpu extra 对齐。
- noob-se7en 整体认可所有权模型：「The distinction between MLX-owned storage in torch_to_mlx() and scoped borrowing in mlx_call() is much clearer.」，alexnails 最终批准：「all my remaining comments are minor, so approving」。
- CPU float64 导出到 MPS 目标失败 (correctness): yeahdongcn 修复：CPU float64 输入在 mx.cpu stream 构造，_prepare_mlx_export() 对 MPS 目标显式拒绝 float64，并新增 test_mlx_call_multi_preserves_cpu_float64 锁定。
- 桥退出路径的竞态 / 死锁疑虑 (correctness): yeahdongcn 复现并修复，补充借用对象存活保持与负 stride 物化处理。

- norm fallback 的 bias 累加精度回归 (correctness): yeahdongcn 修复为 fp32 累加、末尾一次性舍入，由 test_norm_infer_preserves_input_dtype_with_fp32_parameters 覆盖。
- Stage A CI 依赖集与 diffusion 导入冲突 (testing): diffusion 导入冒烟拆到 test_diffusion_import_isolation.py，Stage A 依赖集保持不变。
- 新增 torchcodec 依赖的疑问 (question): yeahdongcn 说明 torchcodec 用于多模态音视频解码路径，与 cuda、xpu extra 对齐。
- 桥所有权模型设计评审 (design): 无需改动；设计获两位 reviewer 认可后合并。

风险与影响

- 风险：崩溃级风险（DLPack 负 stride）：Torch 2.13 的 DLPack 导入器遇负 stride 视图会 abort 整个进程而非抛异常，tensor_bridge.py 的 _has_negative_stride() 预检 + _export_evaluated_mlx() 的 mx.contiguous 物化是唯一防线；已有 abort-safe 子进程测试覆盖，但未来新增输出路径若绕过 mlx_call_multi 的共享物化边界仍可能漏检。版本硬锁定的升级窗口：SGLANG_USE_MLX=1 只接受 stable Torch 2.13.x 与 stable MLX>=0.32.0，Torch 2.12/2.14 和 rc/dev 全部拒绝；MLX 后续 minor 只有形式化放行、缺每版本回归，Torch 2.14 发布后用户会立即遇到硬失败。双框架流同步边界：_serialized_bridge 的 RLock 只覆盖函数内完成的桥工作，无法串行化函数外对同一 MPS 张量的并发使用 / 修改，契约依赖调用方自律。diffusion fallback 数值变化：norm_infer_native 从手工 fp32 分解改为 F.rms_norm / F.layer_norm，与 Triton 内核存在 1e-5 量级差异（测试容忍 2e-5），严格位级一致场景需留意；该代码路径同时服务 CPU Torch 2.11/2.12。依赖面扩大：srt_mps 新增 torchcodec==0.15.0，安装体积与解析复杂度上升。
- 影响：用户：Apple Silicon 上启用 SGLANG_USE_MLX 的用户必须升级到 Torch 2.13 / MLX 0.32+；未启用用户的 import 路径完全不受影响（懒加载 + 子进程测试保证 sys.modules 无 mlx）；diffusion 用户不再有 MLX norm 加速路径，但该路径本就不可达。系统：零拷贝桥的主要收益将在堆叠 PR yeahdongcn/sglang#7 落地的 Torch 权重与算子穿越中兑现；当前 SRT MLX runner 并未被替换，SGLANG_USE_MLX=1 仍走旧 runner，但运行时门控已收紧。团队：本 PR 确立了 Torch-MLX 桥的 ownership/lifetime 契约，成为 Apple Silicon 路线图（issue #19137）上消除 MlxModelRunnerStub drift 的公共基础；review 环节的精度审计与 CI 依赖集把关提升了该路径的工程门槛。
- 风险标记：崩溃级：DLPack 负 stride 需预检物化，依赖版本硬锁定 Torch 2.13.x，双框架流同步边界依赖调用方自律，扩散 fallback 数值行为变化，新增 torchcodec 依赖，CI 依赖集需保持精简

关联脉络

- PR #7 [MPS] Run SRT with Torch ModelRunner and MLX/Metal operators（yeahdongcn/sglang 堆叠 PR）：本 PR 的桥原语直接由该堆叠 PR 消费；其 body 明确以 mlx-032-torch-213-bridge 分支为基础，目标是标准 Scheduler + Torch ModelRunner + 可选 MLX/Metal 算子替换旧 MlxModelRunner。
- PR #22112 Platform feature reporting 机制（review 讨论中引用，标题为描述性转述）：yeahdongcn 回复 noob-se7en 时引用此 PR 作为「平台负责上报特性支持情况」的机制依

据。

- PR #30181 MlxModelRunnerStub drift 修复（标题不可得，来自 issue 7 的引述）：issue 7 将其列为并行 MLX runner 反复漂移的典型修复，本 PR 的所有权契约路线旨在根除这类 drift。