用小体系核对OpenMM平台、精度设置与运行日志

完整操作教程 · 参考固定 · 往返恢复验收

OpenMM能列出CUDA平台，不等于目标计算已经在指定GPU执行。新环境应依次检查平台、Context实际属性与小体系数值，再运行自己的代表体系。本篇用两粒子谐键作为安装检查，给出可直接计算的初始势能；它不是

运行边界：示例代码需按安装版本和真实项目输入复核；未提供真实生物数据，语法或小样本检查不代表科学结果和性能验收。

OpenMM能列出CUDA平台，不等于目标计算已经在指定GPU执行。新环境应依次检查平台、Context实际属性与小体系数值，再运行自己的代表体系。本篇用两粒子谐键作为安装检查，给出可直接计算的初始势能；它不是蛋白体系、显存或每秒模拟速度基准。

先固定平台与精度合同

在目标OpenMM环境运行并保存python -m openmm.testInstallation的完整输出；它用于检查安装，不证明所有模型兼容。下面显式选择CUDA，不进行自动平台回退。设备编号是CUDA可见设备里的逻辑编号，应与调度器分配一致；不要覆盖已有CUDA_VISIBLE_DEVICES去使用未分配卡。

检查项	本例选择	验收
平台	显式CUDA	Context实际名称一致
精度	mixed	读回Precision属性
设备	DeviceIndex=0示例	与可见设备映射对应
小体系	两粒子、单个谐键	初始势能约0.2 kJ/mol
后续	真实项目代表任务	另测RAM、显存与端到端时间

建立Context并读取实际属性

将代码保存为probe_openmm.py。它只依赖OpenMM及NumPy，不下载力场或结构。质量12 dalton、平衡键长0.1 nm、初始距离0.12 nm、力常数1000 kJ/mol/nm²均为教学值；初始势能由二分之一乘力常数再乘位移平方得到0.2。容差仅用于此小例。

```
import json,numpy as np
import openmm as mm
from openmm import unit as u
print("OpenMM",mm.__version__)
names=[mm.Platform.getPlatform(i).getName() for i in
range(mm.Platform.getNumPlatforms())]
print("platforms",names);assert "CUDA" in names
system=mm.System()
for _ in range(2):system.addParticle(12*u.dalton)
force=mm.HarmonicBondForce()
force.addBond(0,1,0.1*u.nanometer,1000*u.kilojoule_per_mole/u.nanometer**2)
system.addForce(force)
integrator=mm.VerletIntegrator(0.001*u.picosecond)
platform=mm.Platform.getPlatformByName("CUDA")
props={"Precision":"mixed","DeviceIndex":"0"}
assert set(props)<=set(platform.getPropertyNames())
context=mm.Context(system,integrator,platform,props)
context.setPositions([mm.Vec3(0,0,0),mm.Vec3(0.12,0,0)]*u.nanometer)
context.setVelocities([mm.Vec3(0,0,0),mm.Vec3(0,0,0)]*u.nanometer/u.picosecond)
assert context.getPlatform().getName()=="CUDA"
actual={k:platform.getPropertyValue(context,k) for k in props}
assert actual["Precision"]=="mixed"
before=context.getState(getEnergy=True).getPotentialEnergy().value_in_unit(u.kilojoule_
per_mole)
assert np.isfinite(before) and abs(before-0.2)<1e-4
initial_force=context.getState(getForces=True).getForces(asNumpy=True).value_in_
unit(u.kilojoule_per_mole/u.nanometer)
```

```
assert np.allclose(initial_force,[[20,0,0],[-20,0,0]],rtol=0,atol=1e-3)
integrator.step(10)
state=context.getState(getEnergy=True,getPositions=True,getForces=True)
assert np.isfinite(state.getPotentialEnergy().value_in_unit(u.kilojoule_per_mole))
assert np.isfinite(state.getPositions(asNumpy=True).value_in_unit(u.nanometer)).all()
assert np.isfinite(state.getForces(asNumpy=True).value_in_unit(u.kilojoule_per_mole/
u.nanometer)).all()
print(json.dumps({"platform":context.getPlatform().getName(),"properties":actual,
                  "initial_potential_kJ_mol":before,"steps":10},indent=2))
```

用日志区分安装与数值错误

先运行nvidia-smi记录可见设备与驱动，再执行python probe_openmm.py并保存标准输出、错误与退出码。缺CUDA平台先查安装包和插件加载；Context创建失败再查驱动、支持架构、内核编译和临时目录。不要在CUDA失败后静默换CPU，并把成功输出说成GPU验收。

初始势能不符先核单位、位置和力常数，而不是扩大数值容差让程序通过。出现NaN则检查模型和积分设置。这个无约束、无长程静电的小例只检查基本计算路径，不能覆盖PME、约束、复杂力场、自定义力或其他插件，真实任务需要逐项验证其所需能力。

迁移项目时另设科学验收

保存体系构建代码、拓扑、坐标、力场、积分器、温压条件及平台属性。比较同一初始构型的能量与力，用项目认可的误差指标；长时间随机轨迹不要求逐位相同。混合精度及设备变化可能改变浮点舍入，不能凭最终坐标相同与否判断全部方法可靠。

两粒子很快，监控采样可能看不到利用率峰，低GPU利用率也不代表未使用CUDA。设备参与以Context实际平台和属性为入口证据，吞吐评估则用足够长的代表体系、预热后计时，并包含需要保留的I/O；不要用小例耗时估算科研项目速度。

记录主机RAM、显存、CPU参与及同卡并发状态。多Context、不同体系和平台属性可能改变资源占用；先完成单任务，再测获配范围内的并发。环境交付应包含版本、插件、样例输出与项目验证范围，不把testInstallation通过写成所有生产协议已验收。

本例初始两粒子沿x轴相距零点一二纳米，解析力大小为二十kJ/mol/nm、方向相反，也可作为独立的力验收。能量与力使用各自单位和容差，不能拿势能的数值直接与力比较。设备名称记录还应与本次作业可见设备的UUID表对应。

配套练习包

本PDF对应的ZIP包含本文代码块、主题练习模板或合成数据、README和SHA256校验文件。请从同一教程页面下载练习包；先核对校验值，再在独立目录复现。

参考资料

OpenMM平台属性与精度 · 来源资料

OpenMM Context真实平台与状态接口 · 来源资料

OpenMM安装检查 · 来源资料

操作建议：先在合成小样本上验证流程，再依据项目参考、标签合同、存储策略与获配资源验收真实数据。