查清软件是否支持GPU并定位当前耗时步骤

完整操作教程 · 参考固定 · 往返恢复验收

从软件支持、驱动、框架小计算到任务日志逐层验证，判断GPU是否真正适用于当前生信步骤。

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。

“这台服务器有显卡，能不能让生信分析快一点？”要回答这个问题，先明确正在运行哪个软件、哪一步和哪一个实现。GPU只有在程序真正使用受支持的加速路径时才可能带来帮助。把原来的CPU命令搬到带GPU的机器上，不会自动改变它的计算方式。

按软件文档、实际运行环境与任务日志逐层核对。模型权重和使用条件也应对应到具体版本。

先定位当前最耗时的步骤

把完整流程拆成数据读取、参考或特征准备、主要计算、结果汇总。逐阶段记录墙钟时间。比如模型推理支持GPU，不代表前面的数据库搜索也自动使用GPU；某个流程在一部分步骤加速，仍可能被数据读取或其他CPU步骤限制。对于已经运行过的任务，先从工作流报告与日志整理耗时，避免重新运行整个项目只为了猜瓶颈。

然后到软件的官方安装、运行参数和性能文档查看“GPU支持”究竟意味着什么。要核对设备类型、框架或编译选项、支持的操作，以及当前命令是否启用了该路径。版本、数据类型和算法模式都可能影响支持范围，不能拿其他工具的GPU用法推断本工具。

核对层次	看到什么才继续	还不能证明什么
软件支持	对应版本写明可用的加速功能	不能证明所有步骤都加速
驱动可见	系统识别到约定设备	不能证明Python环境能使用它
框架可用	所用环境能够创建并运算设备对象	不能证明生信模型适配
任务日志	目标软件记录了实际使用设备或加速模式	不能证明比CPU更划算
同输入基准	结果通过检查且总耗时与费用可比较	不能外推到所有数据规模

在真正运行任务的环境里做小检查

先进入实际的conda环境或容器，而不是在另一套Python里检查。如果使用NVIDIA设备，可先查看nvidia-smi；这只是设备与驱动层的观察。对于基于PyTorch的任务，可以运行下面的小计算，既检查可用性，也让一个真实设备运算完成。它不依赖研究数据，但必须已经安装适配的PyTorch构建。

```
import torch

print("torch:", torch.__version__)
print("build cuda:", torch.version.cuda)
available = torch.cuda.is_available()
print("cuda available:", available)
if not available:
    raise SystemExit("[redacted]CUDA[redacted]/[redacted]")
print("device:", torch.cuda.get_device_name(0))
x = torch.ones((2, 2), device="cuda")
value = (x @ x).sum().item()
print("small operation result:", value)
```

这段程序的数值只用于检查运算链路，不是生信性能基准。即使检查通过，仍需用软件官方样例或自己的小输入确认模型、算子和设备架构兼容。如果失败，不要立即更换更贵显卡，应先区分安装了CPU构建、驱动不兼容、容器没有映射设备、模型不支持相关算子等原因。

主机内存和显存分别测量

输入加载、样本整理和数据库处理可能主要使用主机内存，模型参数与部分计算使用显存。显存有空余而任务被系统杀掉，仍可能是主机RAM不足；反过来，主机RAM很多也不能直接补足一个必须放在GPU上的张量。先根据报错阶段与监控记录定位是哪一种限制，再比较配置。

对GROMACS等具有不同加速模式的软件，应该读取本次运行日志中实际采用的CPU/GPU分工。官方性能文档讨论了多种任务映射和条件，不能只看到环境里有CUDA就断言某个体系一定适合把所有计算放到GPU。首次验证可以选择保守、明确支持的路径，再逐项调整。

形成设备选择结论

如果主要耗时步骤没有受支持的GPU实现，优先优化CPU、内存、I/O或流程组织。如果有可用实现，先跑通代表性任务并检查结果，再做CPU与GPU的完整耗时比较。记录选择理由，例如“目标推理支持该设备且样例通过”，而不是“GPU一般更快”。

最后记录目标步骤、软件构建、实际设备、样例结果和仍未验证的条件。只有这条运行链路通过检查，后续的性能比较才有意义。

参考与版本核对

从系统设备到分析程序逐层检查

GPU是否能用于某个生信步骤，需要经过多个层次：宿主机识别设备、当前任务被分配设备、容器或虚拟环境能访问设备、驱动与运行时兼容、分析软件包含对应GPU后端、所选参数确实走该后端。任一环节缺失，程序都可能回退CPU或直接报错。先记录设备型号、驱动、CUDA/ROCm运行时、容器配置和软件版本，不要把主机上能运行nvidia-smi视为分析已调用GPU。

先用框架级小操作确认设备可访问，再用目标程序的最小有效输入启动一次。查看程序日志中的设备、内核或加速后端信息，并观察运行期间设备利用和显存变化。某些步骤只将特定计算交给GPU，数据解压、排序、索引、文件写入仍可能运行在CPU或受磁盘限制，因此“GPU在用”不代表整条工作流都由GPU执行。

验证层	检查证据	不能据此推出
系统识别	设备列表和驱动状态	任务账号可访问设备
运行环境	容器映射、框架设备查询	目标软件支持该GPU后端
软件能力	官方支持矩阵、命令帮助和日志	当前参数路径使用加速
目标任务	代表输入实际运行记录	所有步骤都由GPU加速
性能对照	相同输入和参数的CPU/GPU耗时	结果质量与科学结论自动相同

公平比较CPU和GPU运行

选择能代表正式分析的数据量，固定软件版本、参数、过滤条件、输出格式和线程设置。分别记录预热、数据读取、模型加载、实际计算和结果写出的时间。首次运行可能包括编译内核或建立缓存，不能和已热启动的另一方案直接比较。CPU方案也要记录线程数和实际CPU额度，避免用受限的单核任务与整卡GPU任务做不公平对比。

保存任务总耗时、峰值内存、显存峰值、CPU使用率、GPU利用率和磁盘读写情况。如果GPU低利用且磁盘等待明显，先测试本地工作盘；如果显存不足，确认是否有其他用户占卡及batch size；如果CPU满载而GPU空闲，检查软件是否只在部分阶段支持加速。对输出进行一致性检查：结果数量、质量指标、随机性设置和必要的数值差异范围都应符合方法要求。

常见错误不要用扩容掩盖

找不到设备，检查任务分配和容器映射；驱动初始化失败，核对驱动与运行时兼容；框架查询不到GPU，检查对应CUDA/ROCm构建；程序成功但GPU始终不动，检查该步骤是否支持GPU及参数是否启用。错误信息应连同版本、启动命令和设备信息保留。增加显存更大的卡无法修复不支持GPU的软件路径，增加CPU也无法修复驱动不匹配。

在共享节点上，确认GPU是否独占、是否允许MIG或其他切分模式、可用显存是否与整卡容量一致。若多个任务并发，分别测量显存峰值和吞吐，不要按单任务速度简单乘以任务数。对于长时间任务，连续观察设备温度、利用率和错误日志，短暂成功只证明小样本可启动。

把验证记录用于资源选择

资源申请单说明目标软件与步骤、输入规模、GPU支持证据、实际试跑结果、CPU和内存需求、显存峰值、工作盘要求及预期并发。无法确认的字段写待验证。将日志和命令与结果校验值关联，后续软件升级、驱动更新或容器变更后重新进行最小验证。只有目标工作负载完成且输出通过科学验收，才把GPU方案记录为适用于该流程。

性能结论要与结果质量一起记录

对随机算法或浮点计算可能有差异的流程，固定随机种子并记录重复运行变异；若方法无法保证逐位一致，应比较预先约定的统计指标和容差，而不是要求文件哈希完全相同。性能表中标明冷启动或热启动、数据存储位置、设备占用和运行范围。只有比较口径一致，CPU/GPU的耗时差异才有参考价值。

申请资源时把目标步骤、输入规模、实测时间、显存峰值、并发数和输出验收结果一并提交。软件升级、驱动更新或容器重建后，用最小测试重新确认设备可用和结果合理，避免环境变化后仍引用旧的基准。

有些软件只在特定子命令、模型或精度模式下支持GPU。逐项核对该任务的官方说明和日志，保存实际启用的选项；不要从软件项目首页笼统推断某个分析模块也支持GPU。若CPU和GPU结果存在差异，先在小样本上重复运行并评估是否属于随机性、数值精度或参数改变。

对暂不支持GPU的流程，CPU、内存、磁盘和并行策略仍可优化。选择配置时基于真实瓶颈，而不是只看设备名称或峰值算力。

报告中保留CPU与GPU两种方案的完整命令和关键日志。将设备空闲、软件不支持、驱动不兼容和资源不足区分记录，下次环境变动时可快速复核。若没有真实任务试跑，只能说明设备检查通过，不能宣称已验证性能提升。

记录变更与复核结论

计划、报价或运行环境变化时，保存变更前后的版本，并说明调整原因、影响范围和确认人。版本化记录能让课题组在预算复核、任务延期或环境迁移时找到一致依据。若某项仍缺资料，明确写成待确认并设置下一步负责人；不要以推测替代报价、实测或学校制度要求。

实际使用结束后，用台账核对资源释放、数据留存、结果交付和费用账单。确认没有未完成任务后再清理临时目录，保留清理清单和必要校验信息。这样后续课题成员可以复用已验证的工作量估算，也能识别哪些结论只适用于当前软件与数据规模。

复核结论应区分已核实事实、估算值和尚未确认事项。附上数据来源与更新时间，便于下一位项目成员判断哪些数字可以沿用。遇到合同、学校要求或软件官方说明发生变化时，应重新检查相关字段，不把旧记录视为永久有效。

如果试跑没有目标程序日志或正式输入，只能将其作为设备连通性检查。正式算力选择仍须通过代表性分析任务和结果质量验收后确认。

保存完整的运行环境说明和驱动版本，避免其他成员无法复现设备调用检查。

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

PyTorch：CUDA可用性检查 · 来源资料

GROMACS：mdrun性能与硬件加速 · 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。