用同一任务比较CPU与GPU的耗时和费用

完整操作教程 · 参考固定 · 往返恢复验收

用同输入、同计算范围和结果验收组织CPU/GPU基准，再按完整占用与真实计费规则比较费用。

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。

一个GPU样例能运行，只解决了兼容性问题。要决定租CPU还是GPU，还要回答：用同一份输入，得到通过验收的结果，总共需要多少时间和费用。本文用GROMACS短基准的组织方式说明比较方法，也适用于其他已经验证CPU/GPU路径的软件。

这里不会给出某张卡的固定加速倍数。体系规模、算法、软件构建和CPU/GPU分工都会影响结果。命令中的步数与线程数是基准协议的演示参数，不是适合所有体系的生产设置，也不能把短基准当作完成科学模拟。

先确定“相同任务”的边界

准备一份已经验收过的bench.tpr，保留输入校验值、GROMACS版本和生成记录。CPU和GPU测试应保持同一输入、相同计算参数与输出要求。不要一边降低精度、缩短序列或删掉分析步骤，一边将速度变化全部归因于硬件。

若使用外部MPI构建，命令的启动方式需要按安装文档调整。下例针对能直接使用gmx mdrun和thread-MPI选项的环境，并且使用支持的非键相互作用GPU路径；先查gmx --version和gmx mdrun -h，不兼容时不要盲目照抄。

```
mkdir -p benchmark/cpu benchmark/gpu
gmx --version > benchmark/version.txt
sha256sum bench.tpr > benchmark/input.sha256

# ■■■■■■■■■CPU■■■■■■■■■■■■■
/usr/bin/time -v -o benchmark/cpu/time.txt   gmx mdrun -s bench.tpr -deffnm benchmark/
cpu/run   -nsteps 20000 -ntmpi 1 -ntomp 4   -nb cpu -pme cpu -bonded cpu -update cpu

/usr/bin/time -v -o benchmark/gpu/time.txt   gmx mdrun -s bench.tpr -deffnm benchmark/
gpu/run   -nsteps 20000 -ntmpi 1 -ntomp 4   -nb gpu -pme cpu -bonded cpu -update cpu
```

如果软件报告某个选项或体系不支持GPU，先修正协议或选择官方支持的路径，不能把失败退出的短耗时作为加速结果。运行时核对两份日志中实际使用的设备与步数。这个测试只改变一个加速环节，后续如要调整PME、更新或线程，应该建立新的测试组，保留其他条件。

重复测量，并保留启动与稳态的区别

测试前确认没有另一个重任务争用设备或工作盘。可重复若干次，每次使用不同输出目录，记录各次结果而非只挑最快的一次。短任务可能主要消耗在启动、数据加载或初始化上，长任务的稳态性能又不能直接代表按次运行的完整体验。根据你的真实工作方式，分别保留端到端时间和软件性能日志。

比较项	必须保持或记录	如何判断
输入和计算范围	相同文件校验值、参数、步数	任务范围一致后才比较耗时
运行条件	版本、CPU线程、设备与其他负载	差异项可解释且可复现
结果有效性	退出状态、日志、输出完整性与适用检查	失败或不满足方法要求的运行不参加比价
端到端耗时	准备、执行、汇总是否都计入	与实际使用场景保持一致
总费用	实际计费单位与附加资源	计算全部占用，不只看GPU计算片段

结果验收先于性能结论

数值软件在不同设备和并行方式下，输出不一定逐字节完全相同。应按该软件和研究方法采用合适的检查，例如是否正常完成、结构和能量记录是否有明显异常、必要统计是否在可接受范围。不要用不恰当的文件哈希相等要求替代科学验收，也不要以“GPU数值不同很正常”为理由跳过所有检查。

若实际任务带随机性，应记录种子或重复设计，并预先确定比较标准。硬件选择不能代替方法选择；若两种实现计算目标并不一致，先确认是否能够公平比较，再谈费用。

把计费规则放进同一个算式

按实际报价填入每个方案的计算资源费用、所需CPU与内存配套、工作盘、必要环境和传输。若按整期租用，就比较完成同一批任务需要的整期范围；若按时间计费，应包括数据准备、空闲占用和最小周期。不能将CPU方案的完整工作量与GPU方案的一小段推理费用直接比较。

最后列出有效结果、整批完成时间和同范围费用。试跑结论只适用于被测输入与运行条件，保留协议后再扩大规模验证。

参考与版本核对

先确认比较的是同一段模拟

GROMACS CPU/GPU基准首先要固定输入TPR、GROMACS版本、模拟步数、cutoff、约束、精度模式和输出频率。若CPU与GPU分别使用不同的步数或精度，性能数字就不能直接比较。记录TPR校验和，并保存gmx dump或其他可复核的输入参数摘要。GPU路线需要确认本机编译配置包含目标后端、设备已分配给任务，且mdrun日志显示实际使用了GPU；仅有nvidia-smi输出并不够。

选择足够长的基准窗口，避开刚启动时的初始化影响，并保留稳定阶段的ns/day或每步耗时。短基准适合检查命令与设备，不一定代表长模拟的稳态。相同运行至少重复两次，记录每次的节点、设备占用、线程、rank、PME设置和存储位置。若服务器同时有其他作业，注明争用情况；否则噪声可能被误认为硬件差异。

固定条件	记录方式	偏差风险
输入和物理设置	TPR校验和、步数、温压耦合和约束	工作量或科学设置不同
软件环境	版本、编译选项、驱动和精度	GPU后端或内核不同
并行布局	MPI ranks、OpenMP线程、设备映射	线程过量或设备争用
计时范围	预热、稳定阶段、输出频率	启动与I/O开销混入结果
科学验收	能量漂移、温度压力及轨迹检查	速度快但模拟不满足要求

分开看计算、通信和输出

mdrun性能不只由GPU型号决定。查看日志中非键相互作用、PME、更新和通信各自占用，判断哪个阶段在等待。GPU加速后若CPU端PME、邻域搜索或MPI通信成为瓶颈，继续增加GPU算力未必改善整体吞吐。改变PME是否放GPU、rank分布或线程数时一次只变一个因素，并确认该组合得到当前版本支持。

输出频率也会影响文件写入和轨迹体积。基准时使用与正式项目一致的能量、坐标和日志输出间隔；若为测试减少输出，应把它作为不同工作负载单独标记。不要把降低轨迹采样频率获得的速度称为硬件加速，因为科学分析所需的时间分辨率也发生了改变。

用同一验收标准评估结果

对比CPU/GPU模拟时，不一定要求轨迹逐位相同，尤其是浮点顺序或并行路径不同的情况；但要按项目预先约定的物理量和数值稳定性检查。核对能量漂移、温度、压力、约束警告和轨迹帧数，查看是否有NaN、粒子飞散或异常中止。短时间成功不能证明长时间模拟稳定，可增加代表性运行段观察温度、设备错误和工作盘增长。

若GPU更快但模拟质量检查不通过，不能采用该配置。若CPU/GPU耗时接近，先查GPU是否实际工作、任务规模是否太小、是否被I/O限制。保存日志和每种配置的命令行，供复算和审稿回应。

把基准换算成项目计划

用有效稳定窗口的ns/day估算所需模拟时间，再加上体系平衡、重复模拟、检查点写入、轨迹分析和失败恢复。独立重复轨迹是额外科学重复，不能当作同一轨迹续跑。预算中分开计算节点小时、GPU小时、轨迹盘和长期归档。若有多个体系并行，先确认每个任务的GPU显存和CPU布局，不要假定单卡基准可线性扩展到多卡。

最终基准记录附上TPR、完整命令、md.log、性能摘要、硬件/软件环境、轨迹验收和适用范围。之后升级GROMACS、驱动、编译方式或改变体系大小，应重新做短基准。没有覆盖的设置写明待测，不把单次数字作为所有项目的性能承诺。

错误排查与重复运行策略

如果GPU基准比CPU慢，先查看日志确认GPU路径确实启用，再查看设备是否有其他进程、任务规模是否太小、输出是否过于频繁以及CPU端是否成为瓶颈。若设备不可见，核对调度分配、容器映射、驱动和GROMACS编译后端。若出现数值不稳定或模拟异常，先核对物理参数、精度、约束与能量最小化，不能简单归因于显卡故障。

基准运行目录与正式模拟目录分开。每种硬件配置单独保存命令和日志，不要覆盖之前通过的运行。比较ns/day时确认统计窗口一致；短任务计入初始化后可能显得特别慢。用结果检查与性能数字一起决定方案，并说明该基准适用于哪种体系规模和运行参数。

做出可解释的节点选择

若CPU方案价格较低但运行时间较长，结合任务截止期和多重复需求计算总节点小时；GPU方案还要核算设备占用时间和显存可用性。若需要同时做多个独立重复，比较单任务更快与多任务并发之间的吞吐，而不是只看单轨迹最快速度。将估算、假设与实测分列，项目规模或GROMACS版本变化后重新验证。

举例来说，若同一TPR在GPU上显著加速，但模拟输出有能量漂移告警，应先暂停性能结论，核查积分步长、精度和GPU设置；若模拟稳定但速度没有提升，则查看日志中GPU是否承担主要计算。基准表把“性能通过”和“科学检查通过”分成两列，避免前者替代后者。

将试跑的输入、输出和硬件配置一并归档。下次模拟体系或粒子数变化时，先判断是否仍在原基准覆盖范围内；如果显著扩大，重新测量稳定阶段并更新资源估算。报告保留测量日期和节点信息，避免不同代硬件的结果被混作同一配置。

若报告要用于设备采购或课题经费申请，附上实际测试条件和原始日志摘要，注明未覆盖的模拟体系、并发和软件版本。这样比较结果可被复核，也避免短基准数据被误当作长期稳定性能。不同节点的价格和服务范围另按当前报价确认。

基准结果应随TPR文件和md.log一同保存，记录生成日期、硬件节点、运行时长和统计窗口。再次运行或升级环境时，先检查输入体系是否仍一致，再决定沿用旧结论还是重测。不同体系的性能不要直接横向比较。

将资源结论写成有条件的建议：说明已测试的数据规模、软件版本、任务并发和结果验收范围。对于未覆盖的输入或环境变化，标注需要复测。不要把一次短时间成功等同于稳定运行，也不要仅凭设备规格推断正式任务耗时。这样其他同学可以按相同条件复核，并判断何时需要更新资源计划。

对外汇报时附上基准日期和环境信息，并说明速度数据来自代表性短测还是完整任务。这样组内成员在复现时可以知道是否需要重新验证。

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

GROMACS：mdrun性能与硬件加速 · 来源资料

GNU Time：内存资源统计 · 来源资料

GNU Time：命令示例 · 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。