用一个小任务测资源，再确定课题组服务器配置

完整操作教程 · 参考固定 · 往返恢复验收

用可追溯的样例试跑记录墙钟时间、内存与工作盘，再通过正式规模复验确定服务器资源。

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。

小样本试跑的作用，是尽早发现环境不兼容、路径错误和资源瓶颈，并建立可比较的基线。它不能单独证明正式数据会同样快，也不能把小样本峰值直接线性外推。试跑前先明确要验证哪一步以及成功时应产生什么结果。

准备一份能代表正式任务的输入

记录软件版本、参考版本、样本类型、FASTQ或矩阵规模、关键参数和输出选项。尽量选择覆盖不同批次或较大输入的代表样本；只用极小的测试文件能验证命令入口，却无法检验正式数据的内存和磁盘压力。输入清单、代码和参考路径固定后再比较机器或参数。

试跑记录	示例字段	用途
任务身份	run_id、样本、软件与版本	能定位运行环境和日志
资源请求	CPU、内存、作业时限、工作盘	区分申请值与机器标称值
运行结果	开始/结束时间、退出码、最后成功步骤	判断是否完整结束
资源观测	峰值内存口径、空间峰值、CPU时间	用于后续对比，不混用指标
结果验收	报告、矩阵/文件数、日志和校验	避免把“进程退出”当成分析成功

用独立目录运行并保存日志

先确认脚本对当前软件版本有效，并让每次试跑写到新的输出目录。以下是通用包装方式；把示例路径和 run_sample.sh 换成实际项目脚本。GNU Time 的最大常驻集是被计时进程的观测值，不一定涵盖全部子进程或同机其他任务；集群作业的总峰值应优先参考调度器或平台记录。

```
set -euo pipefail
run_id=pilot-001
mkdir -p "benchmark/$run_id"
date -Is > "benchmark/$run_id/started.txt"
/usr/bin/time -v -o "benchmark/$run_id/time.txt" \
  bash run_sample.sh \
  > "benchmark/$run_id/stdout.log" \
  2> "benchmark/$run_id/stderr.log"
status=$?
printf 'exit_code=%s\n' "$status" > "benchmark/$run_id/status.txt"
```

若希望即使命令失败也保存退出码，应调整脚本控制流，确保状态捕获语句能在失败后执行；不要因 set -e 在中途退出而漏记状态。对Cell Ranger等工具，使用该版本官方支持的资源参数，并保留 web summary、运行日志和输出目录清单。

怎么判断试跑结果可用于规划

先核对退出码，再打开报告或关键输出验证样本、维度和预期字段；检查日志中断点、临时空间是否触顶以及任务是否被调度器限时或限额终止。只有目标步骤完整、结果通过项目验收，且输入和参数可追溯，才把这次资源观测用于配置比较。

如果换了参考、软件版本、样本规模、BAM输出选项或并发数，应将其视为新的试验条件。多次测量保留全部结果，解释波动原因，不只挑最短耗时或最低峰值。正式配置的结论应写明测试范围、未覆盖的数据类型和仍待确认的限制。

把试跑记录转成采购或租赁依据

试跑汇总可按“测试条件—有效结果—资源峰值—未覆盖条件”写一行结论。例如某版本、某参考、某一类样本在单任务额度下完成，只能证明这组条件通过；不能据此承诺所有样本或多个并发任务都通过。若结果接近内存/磁盘额度，先保留安全余量并用较大样本复测；若慢在读取或写入，增加CPU未必解决。把确实缺少的资源项标明后，再比较候选配置、租期和总成本，避免为未验证的需求提前付费。

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

GNU Time资源统计 · 来源资料

GNU Time示例 · 来源资料

Linux proc文件系统 · 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。