按比较任务安排并发并核算CPU与内存需求

完整操作教程 · 参考固定 · 往返恢复验收

当一份差异表达比较已经跑通，真正拖慢课题组进度的往往是几十项比较怎样排队。一次开很多R进程可能同时读取整个对象，让CPU更忙却更早耗尽内存。本文以已确定方法与分组的任务为前提，把比较清单、单任务峰值和资源额度转成受控批处理方案，不改变统计设计。 先把比较清单冻结成任务表 每项比较有独立编号，记录输入

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。

当一份差异表达比较已经跑通，真正拖慢课题组进度的往往是几十项比较怎样排队。一次开很多R进程可能同时读取整个对象，让CPU更忙却更早耗尽内存。本文以已确定方法与分组的任务为前提，把比较清单、单任务峰值和资源额度转成受控批处理方案，不改变统计设计。

先把比较清单冻结成任务表

每项比较有独立编号，记录输入对象、身份、检验方法、输出目录和版本。处理组效应应保留样本层面设计，不能因为批处理方便把不同问题合并成细胞级两两检验。清单在正式运行前复核，删除重复任务并确认哪些任务确有研究意义；不建议无目的地计算所有组合来制造大量结果。

任务字段	示例填写方式	验收用途
comparison_id	cmp_A_B	避免结果文件覆盖
input	实际对象路径	确认读取同一版本
identity_1与identity_2	已复核分组	避免方向和对照颠倒
method与参数	与已通过样例一致	使任务可追溯
output	独立任务目录	支持失败重跑与汇总

分别求CPU和内存允许的并发

以下是预算演算，不是Seurat规格：若可用于任务的CPU为12个线程，单任务用4个线程；总内存96 GiB，另保留16 GiB，单任务经验证峰值24 GiB，则CPU允许3项、内存也只允许3项。同时运行上限取更严格条件，再考虑实际I/O与其他用户。未测过单任务峰值前，不把这个算式当正式配置承诺。

```
from math import floor
cpu_available = 12
cpu_per_job = 4
memory_total_gib = 96
memory_reserved_gib = 16
memory_per_job_gib = 24
allowed = min(floor(cpu_available / cpu_per_job),
              floor((memory_total_gib - memory_reserved_gib) / memory_per_job_gib))
if allowed < 1:
    raise SystemExit("■■■■■■■■■■■■■■■■")
print("■■■■■■■■■■■■:", allowed)
```

每个R进程读入完整对象时，不能把对象当作所有进程自动共享一份内存。并行方案需要检查worker数量与数据导出方式。CPU线程和进程数要一起核对，避免每个worker又启动一批数学库线程。预算中只计算计划内任务，不能忽略同时运行的交互会话或其他成员。

先用顺序批处理验证接口

准备已经验证的run_de.R，约定参数依次是输入对象、两个身份和输出文件。它应固定正式方法，并在错误时返回非零退出状态。下面顺序运行两个示例比较，保存独立日志；A/B/C标签和对象路径需按项目替换，计时使用Linux GNU time。

```
from pathlib import Path
import subprocess
pairs = [("A", "B"), ("A", "C")]
for first, second in pairs:
    folder = Path("batch-de") / f"{first}_vs_{second}"
    folder.mkdir(parents=True, exist_ok=True)
    command = ["/usr/bin/time", "-v", "-o", str(folder / "resources.txt"),
               "Rscript", "--vanilla", "run_de.R", "object.rds",
               first, second, str(folder / "result.csv")]
    with (folder / "stdout.txt").open("w") as out, (folder / "stderr.txt").open("w") as
err:
        result = subprocess.run(command, stdout=out, stderr=err, check=False)
    (folder / "exit.txt").write_text(str(result.returncode))
```

示例身份被用于目录名，因此真实标签包含路径分隔符或特殊字符时，应改用经过检查的独立任务ID。这里只实现顺序启动，便于先检验脚本参数和日志；不是一个已经提供资源隔离的调度器。顺序样例通过后，再由合适工具实现预算内并发。

小并发试跑后再扩大批次

先选择两项代表性比较重叠运行，观察是否出现比单任务更高的读取与内存压力，检查工作盘和对象版本是否相同。若吞吐量未改善，先查存储争用、重复加载或CPU线程过度订阅。并发越多不必然越快，正式上限应来自有效输出和整体完成时间，而不是CPU曲线更满。

每项完成后核对结果、记录成功或失败；重跑只针对未通过的任务，使用新的运行编号，不能把旧结果误当这一次成功。汇总时按comparison_id连接，而不是按目录遍历顺序拼接。最终应能找到每份结果对应的命令、输入和资源日志。

若批量比较有明确截止日期和持续资源需求，可把任务清单、单任务峰值、预期并发与租期交给云生信评估CPU服务器。资源额度与环境部署按约定提供；差异表达方法、结果复核或统计解释是否需要另行支持，应在咨询时分开说明。

参考资料

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

Seurat FindMarkers官方参考 · 来源资料

GNU time手册（Debian发布） · 来源资料

Linux cgroup v2文档 · 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。