用任务清单与joblog管理批处理、并发和失败重跑

完整操作教程 · 参考固定 · 往返恢复验收

批量运行生信命令时，joblog能保存任务顺序、运行时间和退出状态，但失败重跑依赖输入顺序、命令和日志的一致性。先把任务清单固化，再控制并发和逐样本验收。

运行边界：恢复任务要求清单、序号、命令和joblog对应；输入或参数变化时创建新批次。

先把单样本命令变成可重复任务

批量处理前，先让一条样本命令在独立目录成功运行，并确认输入文件、输出文件、日志和退出码。命令应显式接收样本ID和输入路径，不依赖当前工作目录中的隐式文件。脚本要在失败时返回非零退出码；若命令失败仍返回0，GNU parallel会把失败任务记成成功，之后无法按joblog准确重试。每个任务输出写到样本独立目录，避免多个进程同时覆盖同一个文件。

接着生成不可变任务清单，按固定顺序列出样本ID、输入路径、输入摘要和输出目标。不要让任务清单在运行过程中被脚本追加或排序变化。GNU Parallel的joblog会记录序号、启动时间、运行时间、退出码、信号和实际命令等信息。若想恢复，输入序列、命令和joblog必须保持匹配；官方文档特别指出只按序号匹配时，改变输入或命令可能导致错误任务被跳过或重跑。

并发数应由每个样本CPU、内存、磁盘IO和外部工具线程数共同决定。若每个任务内部已经开8线程，再启动16个并行任务，可能产生128个计算线程和远超申请的内存。可先从-j 1或较小并发开始，按代表性样本测量最大RSS与耗时，再逐步提高。对于需要独占GPU、较大内存或不同优先级的任务，单纯用GNU Parallel的slot数量并不构成资源隔离，仍应使用机构的调度器。

准备内容	为什么要固定	失败后如何处理
样本清单与顺序	joblog按任务序号对应输入	新清单使用新joblog
命令及参数	避免重试了不同分析	参数变更后另建批次
joblog路径	保留每次运行状态	备份后用resume-failed
逐样本输出目录	防止并发覆盖与混淆	验收完整后标记完成

运行joblog并限制并发

GNU Parallel命令可把样本ID作为占位符，将joblog写到批次目录。批次目录应包含清单、命令模板、软件版本、资源配置、开始时间和日志。joblog与stderr日志分开保存；stderr可能包含文件路径和样本信息，要按项目权限保护。每个批次使用独立输出目录，不要把第二轮失败重试输出覆盖到第一轮未经检查的结果上。

--joblog记录的是命令运行，不是科学结果质量。成功退出码只能说明程序认为运行成功，还要验证输出文件存在、大小合理、格式可读、样本ID和输入对应、必要字段齐全。某些工具即使中间步骤失败仍会输出文件，不能以“有文件”作为成功依据。建立轻量检查程序逐条读取输出，失败时写出明确的样本ID与原因。

失败时选择合适的恢复选项

`--resume`会从最后未完成的任务继续；`--resume-failed`会重试失败任务并继续未完成任务；`--retry-failed`仅按joblog重试失败命令，并忽略当前命令行中提供的命令。具体行为与GNU Parallel版本相关，应查看当前安装版本手册。恢复前先备份joblog，检查失败原因是否已修复，并验证输入文件摘要、清单顺序和命令完全一致。若只是某个样本输入损坏，先修复该输入并记录新摘要；如果样本列表或参数发生变化，创建新批次和新日志，不要用旧joblog续跑不同任务。

若任务被Ctrl-C、节点中断或磁盘写满，joblog可能存在未完成项或半截输出。检查joblog最后几行、运行进程和结果文件状态；必要时将未完成输出移到隔离目录。不要直接删除joblog或手工改退出码以便“全部变绿”。重试次数也要有限制，外部服务、网络挂载或许可错误可能持续失败，反复重跑只会消耗资源。

示例：固定输入文件和joblog

以下示例用合成样本ID展示命令结构，不执行任何生信工具。正式运行前把sample_ids.txt写成固定顺序，检查重复ID、缺失输入和路径转义；清单内容改变后创建新的批次目录。

```
set -euo pipefail
mkdir -p run-001/logs run-001/results
sha256sum sample_ids.txt > run-001/sample-list.sha256
parallel --jobs 2 --joblog run-001/joblog.tsv \
  'analysis-command --sample {} --out run-001/results/{} > run-001/logs/{}.out 2> run-
001/logs/{}.err' \
  :::: sample_ids.txt
# Review exit values and validate each per-sample result before accepting the batch.
# After fixing a failure, keep the same sample list, command, and joblog:
parallel --jobs 2 --resume-failed --joblog run-001/joblog.tsv \
  'analysis-command --sample {} --out run-001/results/{} > run-001/logs/{}.out 2> run-
001/logs/{}.err' \
  :::: sample_ids.txt
```

示例中的analysis-command和路径都是占位符；真实命令必须正确引用空格、特殊字符和变量。命令模板有变化时不要沿用joblog。多个操作者可能同时对同一joblog进行恢复，需用锁或由单一负责人执行，避免两个进程重复跑同一序号。日志目录至少保留任务清单摘要、stderr、软件版本和输出检查报告。

汇总失败原因并规划CPU资源

批次结束后按退出码统计成功、失败、信号终止和未运行任务，结合每个样本耗时识别异常长任务。若失败集中在同类大样本，可能是输入规模或内存峰值差异；若失败随机发生，可能是共享存储或节点稳定性。耗时分布不均时，调小任务并发或按文件大小拆分批次可能比增加CPU更有效。GNU Parallel的输出、stderr和joblog本身也会占空间，海量任务注意日志文件数量和归档策略。

CPU服务器配置要根据每任务线程、并发数、峰值内存、样本数量和交付时限估算。申请资源时提供代表性样本的运行记录，而不是只提交样本总数。租用额外CPU环境只解决可用算力和运行空间问题，批次清单、错误处理、命令验证和结果解释仍由项目流程负责。

输入清单和命令要适合安全地重跑

不要将任意文件名直接拼进shell命令。样本ID最好使用受控字符集，并将路径放在任务表的独立列，脚本明确引用。若文件名含空格或特殊字符，GNU Parallel的引用规则需依据实际命令验证；可以用`--null`配合NUL分隔输入，或让程序从固定格式TSV读取并自行处理。运行前检查每行字段数、样本ID唯一、输入可读、输出路径不存在或允许覆盖，避免一条坏记录拖垮整批任务。

joblog中保存完整命令可能泄露路径和参数。把joblog放在项目受控目录，限制权限，归档前脱敏。命令行不能直接包含密码、API token或云凭证；用安全凭证机制提供运行时授权。日志中保存每个样本的stdout/stderr时，同样检查其是否包含样本注释或敏感字段。只向项目成员开放对应批次目录，不要把joblog上传到公开代码仓库。

GNU Parallel的恢复依赖序号。即使输入内容相同，若顺序改变，也可能让旧joblog对应到错误样本。因此在恢复前重新计算清单摘要，并逐行比对当前文件；保存命令模板摘要和Parallel版本。新一轮分析、参数改变、样本增删都创建新run_id。批次日志已完整时，用工具生成成功失败摘要，保留原始joblog只读归档。不要删掉失败行后再将joblog冒充原始记录。

输出验收可以分层执行：先确认每个预期样本都有退出码为0的任务记录，再检查关键文件可读和大小非零，最后由工具特定检查程序验证格式和字段。若输出已有但joblog显示失败，应把文件放到隔离位置并判断是否是部分结果。重试时可以写入attempt目录，最终验收成功后再更新“current”链接，避免并行重跑覆盖旧结果。

在合成任务上演练joblog恢复

正式批处理之前，建议用几条合成输入演练一次成功、一次失败和一次中断。故意让某个合成任务返回非零，确认joblog确实记录退出码，再修正测试命令并使用`--resume-failed`验证只重跑失败和未完成项。不要把这类演练放在真实结果目录。测试结果要检查序号和输入ID映射，确认日志没有重复覆盖成功任务。演练同时验证parallel版本是否支持预期选项、作业系统的信号如何传递以及中断后临时输出如何处理。

如果任务清单由find或数据库查询临时生成，先把输出固化到批次目录并排序，保存文件摘要和生成规则。不要在恢复时重新运行查询，因为文件增删可能改变序号。命令中使用shell变量时，在GNU Parallel的引用环境下确认变量由本地shell还是远端shell展开；引用错误可能让每个任务拿到同一个输入或空参数。先运行`--dry-run`检查展开后的命令，再用一两个合成样本实际执行。

高并发会增加打开文件数、临时目录数量和共享盘随机IO。观察系统限制、文件描述符、inode和网络存储延迟。若任务本身调用多线程程序，应在命令中限制其内部线程，或者相应降低parallel作业数。磁盘空间也要按每个并发任务的临时输出估算，避免所有任务同时写临时文件而占满工作盘。对长时间任务可按批次分组，降低故障时需要重新检查的范围。

批次交付前生成一份汇总表：总任务数、成功数、失败数、未完成数、退出码分布、输出验收通过数、耗时范围、软件版本、输入清单摘要和joblog位置。未通过的样本明确列出失败阶段和后续责任人。若输入问题导致部分任务无法执行，报告中保留这部分，不要只交付成功样本并让接收人误以为批量任务完整完成。每个结果目录写入manifest，避免后续拿错批次。

避免重试产生重复的科研结果

重试命令最好做到幂等：相同输入和参数重复运行时，不会把统计表重复追加、覆盖已验收结果或触发外部副作用。可让每次attempt写入单独临时目录，程序成功且验收后再把结果标记为当前版本。对于无法幂等的脚本，在命令前检查目标文件和运行状态，并由负责人决定是否清理部分产物。不要仅凭joblog显示失败就立即重新提交，因为程序可能已完成主要计算但在最后写日志阶段退出。

GNU Parallel的joblog主要追踪命令执行，不负责解释输出语义。样本清单摘要、代码版本、工具版本、参考库版本和参数应放在同一批次manifest。若更换失败样本输入，保存新SHA并创建新批次；旧joblog保留用于说明之前发生了什么。这样在论文返修或错误追踪时，能区分第一次失败、修改后的重跑和最终验收结果。

每批任务设置明确的验收阈值

对于每种输出格式，预先确定最低验收标准，如文件可解析、样本标签匹配、必要列存在、记录数在合理范围、关键统计量不是缺失值。阈值由分析方法和数据类型决定，不能使用“文件大小大于零”作为唯一标准。失败样本进入隔离表，注明是否需要更换输入、降低并发、增加内存或联系工具维护者。对于不可重复随机算法，运行manifest还要保存种子和软件版本。

若批次规模很大，分成可审计的小批次并分别记录joblog摘要。每个批次结束后确认成功数量与输入清单一致，再开始下一批；这样磁盘空间或软件问题更容易定位。归档时保留失败日志一段明确期限，失败解决后将旧批次标记为历史，不与新结果混在同一目录。

配套练习包

本PDF对应的ZIP包含本文代码、主题模板或合成数据、README和SHA256校验文件。运行前核对包内清单，并按文章说明在独立目录验证。

参考资料

GNU Parallel 官方手册 · 来源资料

GNU Parallel 教程 · 来源资料

util-linux flock 手册 · 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。