为RFdiffusion设计任务建立队列、资源记录与失败重试

完整操作教程 · 参考固定 · 往返恢复验收

RFdiffusion批量任务先按设计目标区分配置与checkpoint，再谈排队。本篇以官方RFdiffusion原始模型的无条件单链骨架生成为基线，不混入RFdiffusion2或RFdiffusion3接口。先用固定长度、一个设计完成环境与输出验收，再扩展队列；这不能保证得到可折叠或有功能的蛋

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。

RFdiffusion批量任务先按设计目标区分配置与checkpoint，再谈排队。本篇以官方RFdiffusion原始模型的无条件单链骨架生成为基线，不混入RFdiffusion2或RFdiffusion3接口。先用固定长度、一个设计完成环境与输出验收，再扩展队列；这不能保证得到可折叠或有功能的蛋白。

建立最小任务，而不是直接扩批

安装环境须在本机驱动、PyTorch、SE3等依赖组合下通过官方示例。官方旧环境文件不能自动保证适合所有现有GPU。Base checkpoint用于本例任务；motif scaffolding、序列固定、对称或二级结构约束可能需要不同模型及配置，必须按官方适用范围确认。

项目	本例	为何保留
contig	100-100	固定长度骨架教学任务
num_designs	1	先测完整单设计
design_startnum	清单中的唯一整数	确定输出编号；确定性模式也用其作种子
output_prefix	每任务独立design	防止旧PDB被cautious跳过

一个获配GPU上顺序运行

jobs.tsv无表头，三列为任务名、固定长度、设计编号，用TAB分隔；例如base001、100、101。RF_REPO和CKPT使用无空白绝对路径，CUDA_VISIBLE_DEVICES由调度器或实际分配设置，不照抄他人的GPU编号。代码只接受本例的一条无条件链，不接受未审核的任意Hydra参数串。

```
set -euo pipefail
: "${RF_REPO:?■■■■commit■RFdiffusion■■■■}"
: "${CKPT:?■■■■■■■■Base checkpoint■■■■}"
test -s "$RF_REPO/scripts/run_inference.py"
test -s "$CKPT"
[[ "$RF_REPO$CKPT" != *[[:space:]]* ]]
mkdir rfd-batch-001
root="$PWD/rfd-batch-001"
git -C "$RF_REPO" rev-parse HEAD > "$root/commit.txt"
test "$(cat "$root/commit.txt")" = "86507b6538f51fce57b5a72477165f03999ed7ae"
sha256sum "$CKPT" > "$root/checkpoint.sha256"
python -c 'import torch; assert torch.cuda.is_available(); print(torch.__version__,
torch.cuda.get_device_name(0))' \
  > "$root/device.txt"
declare -A seen_indexes=()
completed=0
while IFS=$'\t' read -r job length index || [[ -n "$job$length$index" ]]; do
  [[ "$job" =~ ^[A-Za-z0-9_-]+$ ]]
  [[ "$length" =~ ^[1-9][0-9]*$ ]]
  [[ "$index" =~ ^[1-9][0-9]*$ ]]
  [[ -z "${seen_indexes[$index]+used}" ]]
  seen_indexes[$index]=1
  out="$root/$job"
  mkdir "$out"
  /usr/bin/time -v -o "$out/run.time" python "$RF_REPO/scripts/run_inference.py" \
    "inference.ckpt_override_path=$CKPT" "inference.output_prefix=$out/design" \
    inference.num_designs=1 "inference.design_startnum=$index" \
    inference.deterministic=True inference.write_trajectory=False \
```

```
    "contigmap.contigs=[$length-$length]" > "$out/run.log" 2>&1
  test -s "$out/design_$index.pdb"
  test -s "$out/design_$index.trb"
  completed=$((completed+1))
done < jobs.tsv
test "$completed" -gt 0
```

检查最终骨架与运行元数据

从PDB核对链、主链原子、残基数量及坐标是否有限，再检查日志是否正常完成。trb包含实际解析配置、时间及相关映射等信息，需与PDB共同保存；它是pickle数据，只读取自己本次生成且来源可信的文件。PDB非空和trb存在仍不是设计科学质量评价。

确定性模式在当前脚本中按设计编号重置随机种子。不同目录若复用相同编号及相同配置，可能重复生成同一任务，不能当独立重复样本。队列表中为独立设计分配不同编号；重试同一项保留原编号，并另存attempt记录。跨版本、设备或依赖组合不承诺逐位一致。

失败重试保留证据

原始脚本的cautious模式遇到已有PDB可能跳过该项，因此本例要求任务目录全新，同时验PDB与trb。发现只有PDB、没有完整元数据时，不直接补成功标记；保留失败现场，在新目录以相同任务编号重试。重新开始一项设计与从中间扩散状态恢复是两件事，本例没有实现后者。

checkpoint形状不匹配先核任务与模型，输入约束错误先核contig和结构编号，CUDA或SE3错误先核环境，显存不足再看实际长度与作业占用。不要任意缩短扩散步骤后宣称同质量加速。长度、约束与采样设置改变会改变任务，应另列比较。

记录单项完整墙钟、主机RSS、GPU峰值显存和工作盘。GNU time不提供GPU显存峰值，需要配合设备或框架监控。先顺序测代表任务；多个任务共享一张卡时单项时间和总吞吐都要记录，不能从GPU利用率一个数决定并发。后续序列设计、独立结构验证与功能筛选分别登记，避免把骨架生成数量当成功设计数。

相同任务重新排队时同时核对checkpoint校验值，模型文件名相同并不足以确认内容相同。

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

RFdiffusion官方环境、任务与模型说明 · 来源资料

实际设计编号、种子与产物实现 · 来源资料

当前基础配置 · 来源资料

RFdiffusion固定版本官方README · 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。