用代表性数据试跑并规划组装服务器与工作盘

从一个有明确输入与输出的Trinity试跑开始，记录阶段资源和组装验收，再规划全量运行及工作盘。

Trinity 组装前，先用代表性样本确认输入格式、预处理方案和资源申请是否匹配。短试跑可以发现文件名、配对和参数问题，但不能按耗时简单乘以样本量来估算完整组装：数据复杂度、转录本多样性和阶段性峰值都会影响运行。更稳妥的做法是先核对 reads，再用受控的小数据试跑，按日志和作业记录逐步确定内存、线程与临时空间。

先确认读段和样本信息

记录 Trinity 版本、输入文件校验值、单双端类型、读长、样本名及预处理步骤。双端数据的左右文件必须来自同一批样本且顺序对应；多个文件按官方要求以逗号分隔。确认压缩格式可读取，并抽查文件头、记录数和质控报告。链特异性文库要根据建库方案确认方向，不能为提高组装效果随意猜参数。多个生物学重复若要分别组装或合并，先明确后续分析目标并保留样本关系。

短试跑主要回答什么

检查点	记录内容	不能据此推断
输入解析	日志中是否正确识别左右端、读数和路径	全量数据一定能完成
资源峰值	作业峰值内存、CPU、临时盘和失败阶段	所有样本的固定需求
组装概况	输出文件、contig 数量和长度统计	生物学完整性已达标

小试数据应按样本和读段配对一致地抽取，保留抽样方法和随机种子。抽样组装用于检查流程可运行和发现资源瓶颈，不适合直接拿来评价低表达转录本是否完整。

内存与线程按阶段调整

--max_memory 是 Trinity 可用于若干步骤的内存建议上限，并非调度器给作业的硬保证；申请量还要覆盖进程开销及同时运行的程序。官方故障说明中，std::bad_alloc 表示可用 RAM 不足；Butterfly 的 Java 堆错误则要结合 CPU 并发和每个进程的堆需求判断。遇到失败先定位阶段和错误，再决定降低并发、调整申请资源或按官方说明改变参数。降低 --min_kmer_cov 可减少内存，但低表达转录本可能更碎，应将其视为分析选择并保留参数记录，而不是通用修复。

```
set -euo pipefail
: "${LEFT:?set left FASTQ}"
: "${RIGHT:?set right FASTQ}"
: "${OUTDIR:?set a new output directory}"
: "${CPU:?set allocated CPU count}"
: "${MAX_MEMORY:?set Trinity memory limit}"
test -r "$LEFT" && test -r "$RIGHT"
test ! -e "$OUTDIR"
Trinity --version
Trinity --seqType fq --left "$LEFT" --right "$RIGHT" \
  --CPU "$CPU" --max_memory "$MAX_MEMORY" --output "$OUTDIR" \
  2> "$OUTDIR.run.stderr.log"
```

正式使用前先创建父目录，并确保输出路径本身不存在；若要保存标准错误日志，可将日志放在输出目录之外，因为 Trinity 会创建输出目录。运行期间通过调度器记录峰值资源，结束后检查退出码、最终日志和组装统计。不要只以目录存在或任务进程结束作为成功判据。

验收组装结果并保留可复现信息

至少保存版本、完整命令、输入校验值、作业资源、日志、输出路径和 TrinityStats.pl 统计。比较不同参数时固定输入与软件环境，一次只改一个因素；同时检查 contig 长度分布、预期转录本或 BUSCO 等适合课题的质量指标。质量指标要与研究目标对应，不能用 contig 总数单独证明组装更好。若重跑需使用新输出目录或遵循官方恢复说明，先确认旧目录里的中间结果属于同一输入和版本，避免混入其他运行结果。

参考资料

Trinity Running Trinity 官方说明 · 官方资料

Trinity Errors during run 官方排障 · 官方资料

Trinity 降低内存占用说明 · 官方资料