用固定输入测试并发组合并选择租赁资源

完整操作教程 · 参考固定 · 往返恢复验收

固定BAM与排序参数，用有界子进程并发记录每个样本状态和全批时间，再独立验收结果。

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。

选择CPU资源，应比较同一批有效任务在不同并发组合下的总表现。一个样本加线程变快，不说明更多样本一起运行也更快。先固定输入、方法与验收条件，分别记录每任务资源和全批完成时间，再判断瓶颈。

先冻结清单和比较范围

本文以已安装samtools的BAM坐标排序为例，samples.csv含sample,bam两列表头，使用唯一简单ID和实际输入路径。保留输入校验值，测试期间不改文件。不同组合采用新结果目录，同一版本、相同-m与压缩设置，不读取前一轮结果冒充新输出。

下面保存为compare_sort.py。jobs控制同时启动的外部任务，threads传给sort的排序压缩线程参数；它们都必须是正整数。Python线程池只负责等待独立子进程，不把Python解释器线程数当作samtools计算线程。

```
import csv, re, subprocess, sys, time
from concurrent.futures import ThreadPoolExecutor
from pathlib import Path

table_path, output, jobs_text, threads_text = sys.argv[1:5]
jobs, threads = int(jobs_text), int(threads_text)
assert jobs > 0 and threads > 0
with open(table_path, encoding="utf-8", newline="") as handle:
    rows = list(csv.DictReader(handle))
assert rows
ids = [row["sample"] for row in rows]
assert len(ids) == len(set(ids))
assert all(re.fullmatch(r"[A-Za-z0-9_-]+", key) for key in ids)
for row in rows:
    row["bam"] = str(Path(row["bam"]).resolve(strict=True))
    assert Path(row["bam"]).is_file()
assert len({row["bam"] for row in rows}) == len(rows)
root = Path(output).resolve()
root.mkdir()
(root / "tool-version.txt").write_bytes(
    subprocess.check_output(["samtools", "--version"])
)

def run(row):
    start = time.perf_counter()
    folder = root / row["sample"]
    folder.mkdir()
    result = folder / "sorted.bam"
    with (folder / "task.log").open("w", encoding="utf-8") as log:
        command = ["samtools", "sort", "-@", str(threads), "-m", "256M",
                   "-o", str(result), row["bam"]]
        code = subprocess.run(command, stdout=log, stderr=log).returncode
        if code == 0 and (not result.is_file() or result.stat().st_size == 0):
            code = 1
            log.write("VALIDATION_ERROR: missing or empty sorted BAM\n")
        if code == 0:
            code = subprocess.run(["samtools", "quickcheck", str(result)],
                                  stdout=log, stderr=log).returncode
    return row["sample"], code, time.perf_counter() - start

start = time.perf_counter()
with ThreadPoolExecutor(max_workers=jobs) as pool:
    results = list(pool.map(run, rows))
with (root / "summary.csv").open("x", encoding="utf-8", newline="") as handle:
    writer = csv.writer(handle)
    writer.writerow(["sample", "exit_code", "sort_and_check_seconds"])
    writer.writerows(results)
print("batch_seconds", time.perf_counter() - start)
raise SystemExit(any(code != 0 for _, code, _ in results))
```

先在获配额度内比较少量组合

从项目根目录执行下例，输入清单可先选两个代表样本，所有输出目录应不存在。1、2只是比较方法演示，不是推荐服务器规模；每任务256M为sort近似每线程内存设置，不是整个进程或批次硬上限。

```
set -eu
python compare_sort.py samples.csv trial-j1-t1 1 1
python compare_sort.py samples.csv trial-j2-t1 2 1
python compare_sort.py samples.csv trial-j1-t2 1 2
```

sort命令失败或成功后缺文件，都会记录非零状态；批次有任何必要样本失败就返回失败。quickcheck只验证部分文件结构，不逐条读取全部记录。资源比较前还应检查输入输出记录数、排序及项目所需质量，不能只看summary里是零。

记录	比较口径	用途
batch_seconds	本轮全部排序与基础检查窗口	看总批次有效吞吐
每样本秒数	含sort与quickcheck	判断慢样本，不冒充纯排序时间
平台CPU/RAM	全作业连续记录	识别实际重叠与峰值
工作盘和I/O	临时块、结果与共享等待	判断是否应降低并发
全部样本状态	失败原因及内容验收	失败批次不参与成功速度排名

避免顺序与缓存影响结论

以新目录重复，交替组合顺序，记录缓存、节点同时负载和存储位置。第一次读远程输入与已被缓存的重复读取不是同一条件，不清空系统缓存影响其他用户，也不只挑最快的一轮。样本分布变大时重新测量，不能线性外推短样例。

若子进程根本无法启动、工具缺失或系统错误，保留异常和已生成的独立日志，这轮应判为失败并查环境；没有完整summary也不能当没有失败。先用可运行环境确认控制流，再开展正式资源测量。

按真正限制选择下一步

总CPU超额时减少并发或线程，RAM不足时按每任务实际峰值缩批，磁盘争用时先观察临时目录和读写布局。多个约束一起决定可行组合，不能仅用jobs乘threads等于主机核心数就认为资源刚好。

最终记录可稳定完成的输入范围、全批时间、失败与内容验收、实际RAM和盘高水位。只有有效任务仍有持续吞吐缺口，再据这些证据安排资源；增加核心不会自动修复路径、依赖或错误分析方法。

把线程与任务并发分开计算

jobs表示同时运行的sort子进程数，threads是每个子进程的线程参数。一个组合会同时消耗jobs×threads数量级的CPU活动，但实际峰值还受压缩、I/O等待和程序阶段影响。内存则需把每个活动任务的排序缓存、缓冲和系统余量一起观察。先按平台允许的额度测单任务，再增加jobs；不要在已有共享负载时把主机标称核数当成当前可用核数。

解释速度表时保留失败任务

summary.csv写入每个样本的返回码和时间；只对全部达到同一内容验收标准的组合比较有效吞吐。若一组跑完更快但有样本失败，就不能称作更优配置。再检查样本大小分布，避免一个小BAM构成大部分样本，掩盖少数大文件拖慢队列。可以报告成功样本数/总样本数、失败类型以及成功样本中位耗时。

测量工作盘和输入缓存

排序既读BAM又写临时块和结果文件。远程输入盘、临时盘和输出盘若共享同一存储，多个作业会争用读写。记录每轮路径与文件系统、磁盘剩余空间、I/O等待和作业峰值；不要在同一目录重复用旧sorted.bam当输出。若线程增加没有带来更高批次吞吐，且I/O等待增多，降低并发或改善数据布局可能更有效。

怎样核算有效吞吐

完整试验结束后，按全部任务通过文件和内容验收的样本数除以从首个作业启动到最后一个有效结果完成的时间，计算整批有效吞吐。另报告排队时间和计算时间，避免把节点等待误当软件速度。若测试组合有失败，应将失败样本列出并说明原因；不能从分母中悄悄删掉失败后再比较。不同规模样本混合时，可同时列出每小时样本数和总输入GB，读者能看出批次结构。

停止试验的条件

若连续两轮出现同一类明确问题，比如达到任务内存限制或工作盘写满，应暂停提高并发，先按日志调整单项设置；若结果检查与预期不同，先暂停速度比较。不要为了测出最大jobs而耗尽共享队列资源。试验规模应在管理员授权和课题组时限内逐级扩大，发现显著I/O争用时回到上一稳定组合。

用运行状态解释CPU与并发结果

除任务耗时外，尽可能从调度器或系统监控收集实际CPU利用、最大内存、磁盘吞吐、I/O等待和作业组限制。若jobs从1增加到2后CPU利用明显提高、整批吞吐上升且内存与盘仍有余量，可继续小步测试；若CPU利用不升、I/O等待上升且总吞吐下降，则瓶颈更可能在共享存储。若内存达到额度，优先降低并发或每任务工作集。平台监控字段和采样频率各异，应在记录中写来源。

比较方案需包含稳定性而非只看均值

每个组合建议用相同样本集重复执行，报告中位批次时间、最快/最慢范围、失败数和资源峰值。样本大小差异明显时分别列出小、中、大样本耗时，避免单一平均数掩盖长尾。若某配置快但偶尔失败，课题组可能更适合选稳定完成且峰值留量更大的方案。数据规模或节点型号变化后重新校准，不把一次测得的值写成永久配置。

样本清单与统计汇总的边界条件

实际样本表可能有空行、BOM、逗号出现在路径或缺少bam字段。运行前应显式检查表头、空值、路径重复和不可读文件；不要默默跳过坏行，否则成功数与提交数会不一致。若summary.csv在某任务异常时仍需保留部分结果，应让每个worker将状态原子写入独立文件，最后汇总全部状态；当前示例若某子进程意外抛出Python异常，批次不会生成完整summary，因此应保留控制台堆栈并标记本轮无效。

重复试验时记录环境与节点

不同samtools版本可能有不同参数行为；不同文件系统和缓存也改变速度。每轮同时保存操作系统、samtools完整版本、节点标识、工作盘类型、任务分区、CPU/RAM额度和并发组合。将试验输出以日期与组合名区分，避免两轮覆盖。汇报资源时写明这些条件，否则“两个线程更快”的结论不能在课题组换节点后直接复用。

一个选择并发组合的示例

假设组合A总批次时间短，但其中一个大BAM失败；组合B略慢却所有样本通过，且峰值内存和磁盘均有余量。若课题期限允许，B更适合作为基准，再针对大BAM单独测更低并发或更大工作盘。不要将A的成功样本时间与B的失败样本剔除后结果混算。对比表应保留全部运行，包括失败原因和是否符合公平比较条件。

什么时候才有依据增加租赁资源

当代表性样本在合法可用的CPU、内存和I/O组合下均已完成且内容验收通过，仍无法满足队列期限或课题组并发需求，才形成扩展计算资源的证据。提交样本规模分布、有效吞吐、总耗时、资源峰值、失败状态和现有额度；以需要改善的瓶颈说明增加哪一类资源。若主要瓶颈是共享盘延迟或输入错误，增加CPU服务器并不能直接解决。

将试跑结果变成组内可复用配置

把稳定组合写入组内运行说明，包括适用BAM大小范围、jobs、threads、-m、临时目录要求、剩余空间阈值、结果检查方式和失败后的重试步骤。说明该组合在哪种节点、文件系统和输入上测得，以及何时需要重新校准。新成员先跑一个代表样本并核对日志，再提交整批任务。配置文档应是测量结果的摘要，不是脱离场景的硬件型号推荐。

并发之间公平比较的运行安排

为了减少系统负载差异，试验时记录每轮起止时间、队列等待与节点状态，交替运行候选组合。若一个组合在夜间空闲节点、另一个在白天共享盘运行，不能把差异归因于threads/jobs。每轮输入完全相同且输出独立；完成后核对每个样本的有效状态。无法控制节点负载时，增加重复轮次并报告波动范围，不宣称小幅差别具有普遍性。

并发试跑的最终检查清单

提交前确认清单行数与预期样本数一致、样本ID无重复、BAM均能读取且输出目录全新；运行中保存每任务状态和系统资源；结束后确认所有样本通过结构检查、汇总数正确、输入未被修改。再报告试验组合与环境边界。任何一项未完成，都应把这轮标成未验收，避免后续成员拿不完整结果作为配置依据。

公平性原则

所有组合使用同一批输入和验收，不把不同软件版本、不同临时盘或不同清单的结果放到同一排名。无法固定节点负载时，至少记录负载与运行次序，并增加重复。

留存失败波动

连续运行两次都保留失败和波动，再决定是否扩大并发。

并发完成后的文件核查

抽查每个样本输出路径、文件大小、header与quickcheck结果，并确认所有日志都对应同一输入ID。汇总中任何空状态或重复样本都先查清原因，再报告整批吞吐。若后续还需建立索引，把索引耗时和资源单独记录，避免混进排序阶段。

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

samtools sort：线程、内存与输出 · 来源资料

samtools quickcheck范围 · 来源资料

Python：子进程返回状态 · 来源资料

Python：受控线程池 · 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。