大VCF分区间处理：索引、并发、结果拼接与校验

完整操作教程 · 参考固定 · 往返恢复验收

以VCF的POS定义互不重叠窗口，逐片过滤，并与整段结果做流式校验。

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。

大VCF按区间处理，最容易出现的问题是边界重复和结果缺片。bcftools默认区域重叠语义可能将跨边界记录纳入两个窗口，因此拆分前要明确按POS归属，还是按变异覆盖范围提取。本篇用单条染色体的POS分区，保持所有样本和记录内容，完成同一过滤规则下的拆分、拼接与守恒检查。

先固定输入与区域规则

准备排序、BGZF压缩且CSI索引有效的普通VCF，以及与它同组装的FAI。示例以chr1为目标，实际名称必须与VCF完全一致；GT、样本与参考已完成上游验收。这里将每条记录只归属其POS所在窗口，长变异跨窗口也不会复制；若研究要求所有重叠变异，那是另一种选择任务，不能沿用这个守恒定义。

环节	本例约定	验收
坐标	1-based闭区间窗口	下一片从上一片终点加一开始
归属	regions-overlap=0，只看POS	每条源记录最多进入一个分片
过滤	FILTER为PASS且QUAL至少30	只作示例，正式规则先论证
拼接	同一样本及顺序，分片顺序固定	concat后与同条件整段比较

顺序生成分片并保留清单

将代码保存为split_vcf.py。示例窗口十万碱基不是资源推荐，先用代表染色体试跑。它读取FAI中的实际长度，逐段保存日志、资源与索引；失败立即停止，不将已经成功的一部分直接当作全染色体结果。所有输出在新目录，路径按项目替换。

```
from pathlib import Path
import subprocess,hashlib
vcf="/data/cohort.vcf.gz";chrom="chr1";width=100000
lengths={row[0]:int(row[1]) for row in
         (line.split() for line in Path("/ref/genome.fa.fai").read_text().splitlines())}
length=lengths[chrom];assert length>0
expr='FILTER="PASS" && QUAL>=30'
root=Path("vcf-parts-001");root.mkdir();parts=[]
for n,lo in enumerate(range(1,length+1,width)):
    hi=min(length,lo+width-1);region=f"{chrom}:{lo}-{hi}"
    out=root/f"part-{n:05d}.vcf.gz"
    with (root/f"part-{n:05d}.log").open("w") as log:
        subprocess.run(["/usr/bin/time","-v","-o",str(root/f"part-{n:05d}.time"),
            "bcftools","view","-r",region,"--regions-overlap","0","-i",expr,
            "-Oz","-o",str(out),vcf],stdout=log,stderr=subprocess.STDOUT,check=True)
    subprocess.run(["bcftools","index","-c",str(out)],check=True)
    parts.append(str(out.resolve()))
listing=root/"parts.txt";listing.write_text("\n".join(parts)+"\n")
joined=root/"joined.vcf.gz"
subprocess.run(["bcftools","concat","-f",str(listing),"-Oz","-o",str(joined)],check=
True)
subprocess.run(["bcftools","index","-c",str(joined)],check=True)
def fingerprint(args):
    proc=subprocess.Popen(args,stdout=subprocess.PIPE)
    digest=hashlib.sha256();count=0
    for line in proc.stdout:
        digest.update(line);count+=1
    assert proc.wait()==0,"■■■■■■"
    return count,digest.hexdigest()
expected=fingerprint(["bcftools","view","-H","-r",chrom,
                      "--regions-overlap","0","-i",expr,vcf])
actual=fingerprint(["bcftools","view","-H",str(joined)])
assert expected==actual,(expected,actual)
```

```
print("■■■■■■■■■■",actual)
```

验收的不只是记录总数

上面的哈希比较读取未含头部的数据行，验证拼接后的内容与原始整条染色体同规则提取一致，逐行处理不把整份VCF读进Python内存。另保存输入和输出样本列表并比较，抽查窗口两侧POS及跨边界记录；仅总数相等可能掩盖一处重复和另一处遗漏。

空窗口可以产生只有头部的有效VCF，索引成功并不意味着必须有变异；记录数为零应与源区域和过滤规则核对。索引失败先查压缩、排序和contig头；表达式报错回看字段定义，不能把失败分片跳过后继续拼接。使用BED区间文件时还需注意其零起点半开坐标，不能把本例数字直接改后缀当BED。

在一致性通过后再增加并发

先比较同一输入顺序与少量并发时的总完成时间、最大RSS合计和磁盘等待。每个进程读取同一大VCF时，共享盘可能先饱和；压缩线程也会与外层进程叠加。窗口大小影响任务数量和启动开销，不保证越小越快，改动后仍需保持同一覆盖清单。

参考头、过滤表达式、窗口列表和软件版本都作为协议保存。位点需要规范化或拆多等位时，尽量在固定的上游输入完成；若拆分过程中改变记录表示，原先按POS的校验前提会变化。完成此染色体只证明该范围成功，扩到全基因组应按参考顺序另建完整contig清单，不拿文件名的字典序拼接。

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

bcftools区域语义、过滤与concat · 来源资料

VCF位置和记录规范 · 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。