高错误率长序列基因组数据敏感序列识别并行算法

钟诚[1,2]，孙辉[1,2]

（1. 广西大学计算机与电子信息学院，广西 南宁 530004；
2. 广西高校并行分布与智能计算重点实验室，广西 南宁 530004）

摘 要：为解决现有算法难以有效识别高错误率长序列基因组数据中敏感序列的问题，提出一种 CPU 和 GPU 协同计算识别的并行算法 CGPU-F3SR。该算法通过将基因组数据中的长序列分割为多条短序列，引入布隆过滤机制，以免对分割短序列重复计算；采用 *k*-mer 编码策略并行地提取所有短序列中的错误信息，并提出改进的序列相似度计算模型，以提高识别准确率；采取 CPU 和 GPU 协同并行加速短序列相似度计算，以提升识别效率；进而高效、准确地识别出高错误率长序列基因组数据中的 2 类敏感序列——短串联重复序列和疾病相关序列。在长度为 100～400 kbp 的长序列基因组数据中敏感序列识别的实验结果表明，与其他同类并行算法相比，所提 CPU/GPU 并行算法 CGPU-F3SR 识别准确率和查准率分别平均提升 7.77%和 43.07%，假阳性率平均降低 7.41%，识别吞吐量平均提高 2.44 倍。

关键词：敏感序列识别；过滤；相似度计算；序列比对；并行计算

中图分类号：TP338，TP301

文献标志码：A

DOI: 10.11959/j.issn.1000–436x.2023009

Parallel algorithm for sensitive sequence recognition from long-read genome data with high error rate

ZHONG Cheng[1,2], SUN Hui[1,2]

1. School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China
2. Key Laboratory of Parallel, Distributed and Intelligent Computing of Guangxi Universities and Colleges, Nanning 530004, China

Abstract: To solve the problem that existing algorithms were difficult to effectively identify sensitive sequences in genomic data for long-read with high error rate, a recognition algorithm using hybrid CPU and GPU parallel computing, called CGPU-F3SR, was proposed. Firstly, the long-read in genomic data were partitioned into multiple short-read, and the Bloom filtering mechanism was used to avoid repeated calculation of the short-read. Secondly, the *k*-mer coding strategy was used to extract in parallel the error information of all short-read, the recognition accuracy was promoted by improving the sequence similarity calculation model. Finally, CPU and GPU were used to coordinate and parallel to accelerate the calculation of short-read similarity to improve recognition efficiency. As a result, both two types of sensitive sequences including short tandem repeats and disease related sequences could be identified efficiently and accurately from genome data for long-read with high error rate. The experimental results of recognizing sensitive sequences from genomic data for long-read with length 100～400 kbp each show that, compared with existing parallel algorithm, the average recognition accuracy and precision rate of proposed CPU/GPU parallel algorithm CGPU-F3SR are increased by 7.77% and 43.07% respectively, its average false positive rate is reduced by 7.41%, and its average recognition throughput is increased by 2.44 times.

Keywords: sensitive sequence recognition, filtering, similarity calculation, sequence alignment, parallel computing

收稿日期：2022–08–29；**修回日期：**2022–11–11

通信作者：孙辉，adairmillersh@gmail.com

基金项目：国家自然科学基金资助项目（No.61962004）；广西研究生教育创新计划基金项目资助（No.YCSW2021020）

Foundation Items: The National Natural Science Foundation of China (No.61962004), Innovation Project of Guangxi Graduate Education (No.YCSW2021020)

0　引言

基因组数据是一类重要的医疗大数据，被广泛应用于生物医学、定制化医疗服务、法医学鉴定和全基因组关联研究等领域[1-2]。基因组数据敏感序列是指可用于发起推断攻击、成员身份攻击、身份追踪攻击和完全攻击等基因组数据隐私攻击的核苷酸序列，包括短串联重复（STR, short tandem repeat）序列和疾病相关序列（DRS, disease-related sequence）[3]。不同于网络隐私数据的可重塑性，基因组数据敏感序列信息的泄露会对隐私泄露者的医疗保险、社会就业和公共福利等产生不利影响[4]。通过识别过滤基因组数据中的敏感信息，可以实现基因组数据敏感信息脱敏，有助于降低基因组数据隐私敏感信息受攻击的风险。

基因组数据中的敏感序列包括 STR 和 DRS 这 2 类。敏感序列识别串行算法的设计主要基于 Hash 映射思想，即通过产生序列 Hash 指印来实现敏感序列的精确识别。此类代表算法有 SRF（short read filter）[3]、LRF（long read filter）[4]和 PriLive[5]等。这些基于序列 Hash 指印思想的串行算法对高错误率的长序列基因组数据敏感序列识别准确率较低，且难以满足对大规模长序列实时识别过滤的要求。为此，一些学者基于多核 CPU 集群系统，通过设计多核 CPU 并行算法，在准确识别含有错误信息的敏感序列的同时，加速求解大规模基因组数据敏感序列识别问题。

关于基因组数据敏感序列识别并行算法的研究，文献[6]基于 STR 挖掘工具 FLASH（fast length adjustment of short read）和 PERF（Python exhaustive repeat finder），在 Hadoop 平台上开发了 CPU 并行加速的基因组数据短串联重复序列识别算法 BigFiRSt。文献[7]采用划分后缀分组方法，提出基于 CPU 集群系统的最大串联重复序列识别并行算法。在串行算法 FindTRs 的基础上，文献[8]设计了在多核 CPU 集群系统上运行的大规模短串联重复序列识别并行算法 MPI-dot2dot。根据 STR 的周期重复性质，文献[9]在多核 CPU 集群系统上设计实现了生物序列模式精确匹配的并行算法。文献[10]通过枚举所有具有统计显著性的局部最优出现频率的子串，提出了一种面向多核 CPU 系统的频繁子串模式挖掘并行算法。上述多核 CPU 并行算法仅针对基因组数据中的 STR 进行识别，并没有处理基因组数据中 DRS 的识别，因此这些并行算法仅能识别出基因组数据中的部分敏感序列。对于低错误率的基因组数据，文献[11]提出一种多核 CPU 并行算法，以识别基因组数据中的短串联重复序列和疾病相关序列。

已有的基因组数据敏感序列识别并行算法的设计主要是针对敏感序列 2 种类型（STR 和 DRS）中的一种进行处理的；有的并行算法虽然同时处理 STR 和 DRS 的识别，但它面向的是低错误率序列构成的基因组数据，而不是高错误率序列构成的基因组数据。此外，已有的基因组数据敏感序列识别并行算法是多核 CPU 并行算法，随着单分子实时测序和牛津纳米孔测序等第三代测序技术的发展，测序平台产生的序列具有长度长和错误率高的特点[12]。已有的并行算法难以高效、准确地识别出高错误率长序列构成的基因组数据中的 2 类敏感序列 STR 和 DRS。为解决此问题，本文研究改进序列相似度计算模型，进而提出一种 CPU 和 GPU 协同计算加速识别高错误率长序列基因组数据中 STR 和 DRS 的并行算法 CGPU-F3SR。该并行算法的特色和创新介绍如下。

1) 采用滑动窗口策略和 *k*-mer 编码方法，分割并提取高错误率长序列基因组数据中的核苷酸序列碱基错误信息。引入布隆过滤器，对高错误率长序列基因组数据进行高效过滤，避免算法对重复数据的多次计算。

2) 根据短串联重复序列和疾病相关序列的结构特点，提出一种改进的序列相似度计算模型，实现准确识别出高错误率长序列基因组数据中的 2 类敏感序列（STR 和 DRS）。

3) 对基因组数据测序序列进行周期划分，设计实现 CPU 和 GPU 协同计算的敏感序列识别并行算法，大大提升算法的运行效率。

1　方法

设待识别基因组数据共有 l 条长序列，$N_s[0{:}n_s{-}1]$ 表示长度为 n_s 的第 s 条长序列，s=0,1,2,…,l–1。本文提出的 CPU 和 GPU 协同计算的高错误率长序列基因组数据敏感序列识别并行算法包括序列分割过滤、短串联重复序列并行识别、疾病相关序列并行识别和生成掩码核苷酸序列 4 个阶段。

1.1　并行算法 4 个阶段

1.1.1　序列分割过滤

为并行识别高错误率长序列基因组数据中的

敏感序列，首先将长序列 $N_s[0{:}n_s-1]$分割为 n_s-r+1 条、每条长度为 r 的短序列 $R_{s,j}=N_s[j{:}j+r-1]$，其中，短序列 $R_{s,j}$ 和 $R_{s,j+1}$ 共享 $r-1$ 个重叠碱基，$j=0,1,2,\cdots,n_s-r$，$s=0,1,2,\cdots,l-1$，$n_s>r$。因为人类 DNA 序列中约 99.5%的序列具有高度一致性，所以将长序列分割为短序列会产生许多相同的短序列[3]。

布隆过滤器由一个二进制位数组（位向量）和若干 Hash 函数组成，多用于数据去重以及数据检索[13]。为避免对重复短序列的多次计算，本文引入双布隆过滤机制对短序列 $R_{s,j}$ 并行过滤去重，$j=0,1,2,\cdots,n_s-r$，$s=0,1,2,\cdots,l-1$。

设布隆过滤器 BF_1 用于敏感序列去重，其位数组和使用的 Hash 函数数目分别为 $B_1[0{:}b_1-1]$和 h_1。布隆过滤器 BF_2 用于非敏感序列去重，其位数组和使用的 Hash 函数数目分别为 $B_2[0{:}b_2-1]$和 h_2。为了初始化双布隆过滤器，需根据式(1)计算 B_1 和 B_2 的位数组规模 b_1 和 b_2[13]。

$$b_i=\frac{t_i\ln(p_i^{-1})}{(\ln 2)^2},i=1,2 \tag{1}$$

其中，p_i 表示布隆过滤器误报率，t_i 表示 BF_i 可容纳序列数目的上限。通过预设参数 p_i 和 t_i，可以根据式(1)计算得到位数组规模 b_i，$i=1,2$。此时，可以计算 BF_i 使用的 Hash 函数数目 h_i[13]为

$$h_i=\left\lceil\frac{b_i}{t_i}\ln 2\right\rceil,i=1,2 \tag{2}$$

依据式(1)和式(2)计算出双布隆过滤器的参数 b_1 和 b_2 以及 h_1 和 h_2 之后，初始化布隆过滤器位数组 $B_{1,i}=0$ 和 $B_{2,k}=0$，$i=0,1,2,\cdots,b_1-1$，$k=0,1,2,\cdots,b_2-1$。

当初始化双布隆过滤器完成后，将短序列 $R_{s,j}$ 进行周期划分，然后使用 Pr 个 CPU 核心通过双布隆过滤器 BF_1 和 BF_2 并行地对短序列 $R_{s,j}$ 进行过滤去重，$s=0,1,2,\cdots,l-1$，$j=0,1,2,\cdots,n_s-r$。

为方便理解短序列周期划分思想，图 1 给出了块划分和周期划分短序列的示例。

人类基因组数据测序序列中存在连续的串联重复区域。若将分割得到的短序列采用图 1(a)所示块划分方式并行处理，则会造成部分线程处于“计算等待”状态。为均衡数据计算，使每个线程达到负载均衡，本文采用图 1(b)所示的周期划分对短序列进行并行处理，并采用双布隆过滤器过滤去重。

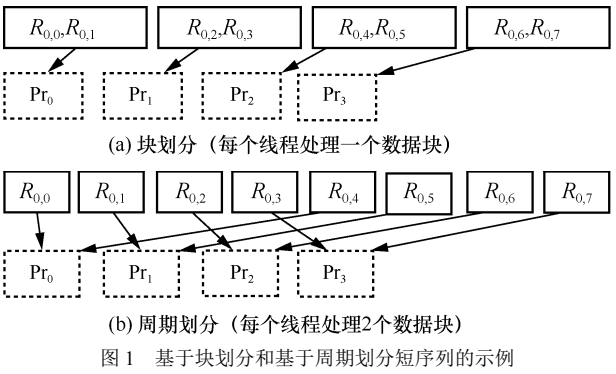

图 1 基于块划分和基于周期划分短序列的示例

本文并行短序列过滤去重方法介绍如下。

Pr 个 CPU 核心并行对 $R_{s,v+d\mathrm{Pr}}$ 执行 h_1 次 Hash 计算，以获得 h_1 个 Hash 值 $H_{1,i}(R_{s,v+d\mathrm{Pr}})$，且对 BF_1 中的位数组 B_1 进行查询。

若 $B_1[H_{1,i}(R_{s,v+d\mathrm{Pr}})]$的值均为 1，$i=0,1,2,\cdots,h_1-1$，则表明 $R_{s,v+d\mathrm{Pr}}$ 是重复短序列且为敏感短序列；若 $B_1[H_{1,i}(R_{s,v+d\mathrm{Pr}})]$的值不全为 1，则需对 $R_{s,v+d\mathrm{Pr}}$ 执行 h_2 次 Hash 计算，以获得 h_2 个 Hash 值 $H_{2,i}(R_{s,v+\ d\mathrm{Pr}})$，并对 BF_2 中的位数组 B_2 进行查询。

若 $B_2[H_{2,i}(R_{s,v+\ d\mathrm{Pr}})]$的值均为 1，$i=0,1,2,\cdots,h_2-1$，则表明 $R_{s,v+d\mathrm{Pr}}$ 为重复短序列且为非敏感短序列；若 $B_2[H_{2,i}\ (R_{s,v+d\mathrm{Pr}})]$的值不全为 1，则表明 $R_{s,v+\ d\mathrm{Pr}}$ 不存在于 BF_1 或 BF_2 中，$R_{s,v+\ d\mathrm{Pr}}$ 不是一条重复序列，此时需对 $R_{s,v+d\mathrm{Pr}}$ 进行短串联重复和疾病相关序列识别处理，$d=0,1,2,\cdots,\left\lceil\frac{n_s-r+1}{\mathrm{Pr}}\right\rceil-1$，$v+d\mathrm{Pr}\leqslant n_s-r$，$v=0,1,2,\cdots,\mathrm{Pr}-1$，$s=0,1,2,\cdots,l-1$。

通过并行地对短序列过滤去重，可避免对短序列的重复计算。若短序列不存在于布隆过滤器 BF_1 或 BF_2，则需对短序列进行短串联重复序列识别的处理。

1.1.2 短串联重复序列并行识别

STR 也被称作微卫星 DNA，由 1～6 个碱基的重复基序组成，约占人类基因组数据的 3%[3]。短串联重复序列的重复结构性质表明，通过合理分割短串联重复序列，可以得到记录短串联重复序列基本重复单元的相同子段[14]。在借鉴文献[14]思想的基础上，本文使用 Pr 个 CPU 核心并行地识别高错误率长序列基因组数据中的短串联重复序列。

并行识别短串联重复序列的方法介绍如下。

首先，Pr 个 CPU 核心并行分割短序列 $R_{s,v+d\mathrm{Pr}}$ 为 4 个长度均为 w 的短片段集合 $W=\{W^i_{s,v+d\mathrm{Pr}}\}$，$i=0,1,2,3$，$v=0,1,2,\cdots,\mathrm{Pr}-1$，$d=0,1,\cdots,\left\lceil\frac{n_s-r+1}{\mathrm{Pr}}\right\rceil-1$，

$v+d\mathrm{Pr}\leqslant n_s-r$，$s$=0,1,⋯,$l$–1。

然后，Pr 个 CPU 核心依据 k-mer 编码参数 k 和核苷酸字母表Σ={A,C,G,T}并行地生成 4^k 个 k-mer，且统计短片段 $W^1_{s,v+d\mathrm{Pr}}$ 和 $W^2_{s,v+d\mathrm{Pr}}$ 中每个 k-mer 出现的频数，将 $W^1_{s,v+d\mathrm{Pr}}$ 和 $W^2_{s,v+d\mathrm{Pr}}$ 转换为记录 k-mer 频率的向量 $\vec{\boldsymbol{W}}^{1}_{s,v+d\mathrm{Pr}}$ 和 $\vec{\boldsymbol{W}}^{2}_{s,v+d\mathrm{Pr}}$，$v$=0,1,2,⋯,Pr–1，$d$=0,1,2,⋯,$\left\lceil\frac{n_s-r+1}{\mathrm{Pr}}\right\rceil-1$，$v+d\mathrm{Pr}\leqslant n_s-r$，$s=0,1,\cdots,l-1$。

最后，Pr 个 CPU 核心并行计算短序列 $R_{s,v+d\mathrm{Pr}}$ 局部片段 $W^1_{s,v+d\mathrm{Pr}}$ 和 $W^2_{s,v+d\mathrm{Pr}}$ 的相似度 sim($W^1_{s,v+d\mathrm{Pr}}$, $W^2_{s,v+d\mathrm{Pr}}$)，且根据短串联重复序列相似度阈值 str_sim 判断 $R_{s,v+d\mathrm{Pr}}$ 是否为短串联重复序列。若长序列 N_s 分割出的短序列 $R_{s,v+d\mathrm{Pr}}$ 为短串联重复序列，则置结果向量中 res[v+$d$$r$]的值为 3，$v+d\mathrm{Pr}\leqslant n_s-r$，$d=0,1,\cdots,\left\lceil\frac{n_s-r+1}{\mathrm{Pr}}\right\rceil-1$，$s$=0,1,⋯,$l$–1，$v$=0,1,2,⋯,Pr–1。

为降低将非敏感序列误识别为短串联重复序列的概率，根据短串联重复序列的结构特点，文献[15]引入控制参数 c_str 和皮尔逊相关系数（PCC, Pearson correlation coefficient）改进序列局部片段相似度的计算为

$$\mathrm{sim}\left(W^1_{s,v+d\mathrm{Pr}},W^2_{s,v+d\mathrm{Pr}}\right)=c\_\mathrm{str}\cdot\frac{\sum_{i=0}^{4^k-1}(\vec{\boldsymbol{X}}-\bar{X})(\vec{\boldsymbol{Y}}-\bar{Y})}{\sqrt{\sum_{i=0}^{4^k-1}(\vec{\boldsymbol{X}}-\bar{X})^2}\sqrt{\sum_{i=0}^{4^k-1}(\vec{\boldsymbol{Y}}-\bar{Y})^2}}$$

$$c\_\mathrm{str}=\begin{cases}1, \max\left(\vec{\boldsymbol{R}}_{s,v+d\mathrm{Pr}}\left[\frac{r}{4}:\frac{3r}{4}\right]\right)\geqslant\left\lceil\frac{2w-k+1}{6}\right\rceil\\ 0,\text{其他}\end{cases}\quad(3)$$

其中，$\vec{\boldsymbol{R}}_{s,v+d\mathrm{Pr}}\left[\frac{r}{4}:\frac{3r}{4}\right]$ 为 $R_{s,v+d\mathrm{Pr}}\left[\frac{r}{4}:\frac{3r}{4}\right]$ 的 k-mer 频率向量，$\bar{X}=\bar{Y}=\frac{w-k+1}{4^k}$，$\vec{\boldsymbol{X}}=\vec{\boldsymbol{W}}^{1}_{s,v+d\mathrm{Pr}}$，$\vec{\boldsymbol{Y}}=\vec{\boldsymbol{W}}^{2}_{s,v+d\mathrm{Pr}}$，$d=0,1,\cdots,\left\lceil\frac{n_s-r+1}{\mathrm{Pr}}\right\rceil-1$，$v$=0,1,⋯,Pr–1，$v+d\mathrm{Pr}\leqslant n_s-r$，$s$=0,1,⋯,$l$–1。

1.1.3　疾病相关序列并行识别

DRS 是指携带疾病易感基因的核苷酸序列[11]。不同于短串联重复序列，疾病相关序列更多体现在位点上的变异，而非结构上的重复。为此，本文通过改进序列相似度计算模型，将待识别序列和第三方数据库中疾病相关序列进行计算比对，以并行识别出高错误率长序列基因组数据中的疾病相关序列。

1) 并行构建敏感序列词典

构建敏感序列词典是为了降低大量短序列与第三方数据库中疾病相关序列相似度计算的代价。本文使用二维数组 hostDict[0:m–1][0:4^k+2]表示 CPU 端构建包含 m 条序列的敏感序列词典。

并行构建敏感序列词典的方法介绍如下。

首先，Pr 个 CPU 核心并行地从第三方数据库提取每个疾病易感基因的染色体编号 CHR_ID 和染色体位置 CHR_POS，并使用 pysam 工具[16]从人类参考基因组 HG38 中提取 m 条长度为 r 的疾病相关序列 $D_{v,i}$，i=0,1,2,⋯,m–1，v=0,1,2,⋯,Pr–1。

然后，Pr 个 CPU 核心依据 k-mer 编码参数 k 和核苷酸字母表Σ={A,C,G,T}并行生成 4^k 个 k-mer，通过统计 $D_{v,i}$ 中每个 k-mer 出现的频数，将 $D_{v,i}$ 编码为记录 k-mer 统计频数的词频向量 $\vec{\boldsymbol{D}}_{v\times i}=\{D_{v\times i,0}, D_{v\times i,1}, \cdots, D_{v\times i,a}, \cdots, D_{v\times i,4^k-1}\}$，其中，$D_{v\times i,a}$ 为 $D_{v,i}$ 中第 a 个 k-mer 的统计频数，$0\leqslant D_{v\times i,a}\leqslant r-k+1$，$a$=0,1,2,⋯,$4^k$–1，$i$=0,1,2,⋯,$m$–1，$v$=0,1,2,⋯, Pr–1。

最后，Pr 个 CPU 核心并行地计算疾病相关序列 $D_{v,i}$ 的 3 个碱基 G、C、A 的碱基含量 $\mathrm{GCA}_{D_{v,i}}=\{G_{D_{v,i}},C_{D_{v,i}},A_{D_{v,i}}\}$，且将$\{\vec{\boldsymbol{D}}_{v\times i},\mathrm{GCA}_{D_{v,i}}\}$赋值给 hostDict[$i$][0:$4^k$+2]，$i$=0,1,2,⋯,$m$–1，$v$=0,1,2,⋯,Pr–1。

2) 疾病相关序列相似度并行计算

在判断短序列 $R_{s,j}$ 是否为疾病相关序列之前，本文已经对 $R_{s,j}$ 进行过滤去重和短串联重复序列识别。因此，仅需要对经上述步骤处理后剩余的 g 条序列进行疾病相关序列识别。将剩余的 g 条序列和敏感序列词典 hostDict[0:m–1][0:4^k+2]中的 m 条序列进行相似度计算比对十分耗时。为此，本文采用多核 CPU 和 GPU 协同并行计算以加速疾病相关序列的识别，j=0,1,2,⋯,n_s–r，s=0,1,2,⋯,l–1。

CUDA（computer unified device architecture）并行编程模型将 GPU 线程层次结构抽象成线程块网格 Grid 和线程块 Block 两层[17]。在 CUDA 线程层次结构中，Block 和 Grid 由 uint3 类型定义的三维向量 blockIdx.x、blockIdx.y、blockIdx.z 和 threadIdx.x、threadIdx.y、threadIdx.z 表示[17]。为准

确定位识别疾病相关序列，本文采用一维线程块网格和线程块进行 CUDA 线程索引。

设参数 gpuBlock 表示 GPU 线程块 Block 的大小，tID 表示 GPU 线程索引，$\mathrm{GCA}_{R_{s,\mathrm{tID}}}=\{G_{R_{s,\mathrm{tID}}},C_{R_{s,\mathrm{tID}}},A_{R_{s,\mathrm{tID}}}\}$表示第 tID 个短序列的 3 个碱基 G、C、A 的碱基含量，词频向量 $\vec{\boldsymbol{R}}_{s,\mathrm{tID}}$ 表示短序列 $R_{s,\mathrm{tID}}$ 的 k-mer 编码核苷酸序列，tID≤g–1，g≥ gpuBlock。

根据相似性序列具有碱基含量相似的特点[15]，本文引入相似性检索控制参数 c_dis，改进计算序列 $\vec{\boldsymbol{D}}_i$ 和 $\vec{\boldsymbol{R}}_{s,\mathrm{tID}}$ 相似度 $\mathrm{sim}(D_i,R_{s,\mathrm{tID}})$ 为

$$\mathrm{sim}(D_i,R_{s,\mathrm{tID}})=c\_\mathrm{dis}\cdot\frac{\sum_{i=0}^{4^k-1}\left(\vec{\boldsymbol{D}}_i-\bar{D}_i\right)\left(\vec{\boldsymbol{R}}_{s,\mathrm{tID}}-\bar{R}_{s,\mathrm{tID}}\right)}{\sqrt{\sum_{i=0}^{4^k-1}\left(\vec{\boldsymbol{D}}_i-\bar{D}_i\right)^2}\sqrt{\sum_{i=0}^{4^k-1}\left(\vec{\boldsymbol{R}}_{s,\mathrm{tID}}-\bar{R}_{s,\mathrm{tID}}\right)^2}}$$

$$c\_\mathrm{dis}=\begin{cases}1,1-|\mathrm{GCA}_{R_{s,\mathrm{tID}}}[e]-\mathrm{GCA}_{D_i}[e]|\geqslant \mathrm{dis\_sim}\\ 0,\text{其他}\end{cases} \tag{4}$$

其中，e=0,1,2，i=0,1,⋯,m–1，tID=0,1,⋯,g–1，s=0,1,⋯,l–1，dis_sim 为疾病相关序列相似度阈值。若 $\mathrm{sim}(D_i,R_{s,\mathrm{tID}})$≥dis_sim，则序列 $R_{s,\mathrm{tID}}$ 是一条疾病相关序列，且它和敏感序列词典中第 i 条序列 D_i 具有很高的序列相似度。

1.1.4 生成掩码核苷酸序列

为保存最终识别结果，本文采用类似于文献[3]的方法，根据 **res**[0:n_s–1]生成测序长序列 N_s[0:n_s–1]的 2 条掩码序列 $N_s^{\mathrm{Pri}}[0:n_s-1]$ 和 $N_s^{\mathrm{Pub}}[0:n_s-1]$，其中，$N_s^{\mathrm{Pub}}[0:n_s-1]$ 表示掩码非敏感序列，$N_s^{\mathrm{Pub}}[0:n_s-1]$ 中的敏感序列片段将以“*”掩码；$N_s^{\mathrm{Pri}}[0:n_s-1]$ 表示掩码敏感序列，$N_s^{\mathrm{Pri}}[0:n_s-1]$ 中的非敏感序列片段将以“*”掩码。

1.2 并行算法描述与分析

设 GWAS Catlog 数据库中疾病相关序列数据集为 disData（包含 m 组数据），HG 表示疾病相关序列并行识别步骤构建敏感序列词典时所使用的参考基因组，并行算法使用的 CPU 核心数为 Pr，使用的 GPU 线程数为 Cu。

图 2 给出了 CPU/GPU 并行算法 CGPU-F3SR 的处理流程。

图 2 中，**res**[0:n_s–1]表示记录每条短序列的敏感序列识别结果的辅助数组，且 res[j]∈{0,1,2,3,4}分别表示原始短序列结果标记、BF_1 过滤标记、BF_2 过滤标记、短串联重复序列标记和疾病相关序列标记，j=0,1,2,⋯,n_s–1，s=0,1,2,⋯,l–1。hostRes[0:g–1]表示保存疾病相关序列的识别结果的辅助数组，其中，$g=\sum_{j=0}^{n_s-r}I\left(\mathrm{res}[j]\right)=0$，$I$(res[$j$])=0 表示指示函数（条件成立则取值为 1，否则取值为 0）。strIndex[0:g–1]表示保存疾病相关序列识别的短序列索引的辅助数组。式(4)中，$\mathrm{GCA}_{R_{s,\mathrm{tID}}}$ 和 $\vec{\boldsymbol{R}}_{s,\mathrm{tID}}$ 的求解需要做许多分支判断。若使用 GPU 计算 $\mathrm{GCA}_{R_{s,\mathrm{tID}}}$ 和 $\vec{\boldsymbol{R}}_{s,\mathrm{tID}}$，则将使部分 GPU 线程处于长时间“计算等待”状态，为均衡 GPU 端数据运算，本文引入辅助数组 hostData[0:g–1][0:4^k+2]，并使用多核 CPU 预编码核苷酸序列和碱基含量统计，以简化 GPU 端数据运算，提升 GPU 端的运行效率。

算法 1 给出了 CGPU-F3SR 的流程。

算法 1 CGPU-F3SR

输入 N_0[0:n_0–1]～N_{l-1}[0:n_{l-1}–1]，r，p_1，p_2，t_1，t_2，k，str_sim，dis_sim，HG，disData，gpuBlock，Pr，Cu

输出 $N_s^{\mathrm{Pri}}[0:n_s-1]\sim N_{l-1}^{\mathrm{Pri}}[0:n_{l-1}-1]$，$N_s^{\mathrm{Pub}}[0:n_s-1]\sim N_{l-1}^{\mathrm{Pub}}[0:n_{l-1}-1]$

1) 使用参数 p_1、p_2、t_1 和 t_2，由式(1)和式(2)计算参数 b_1、b_2、h_1、h_2，并初始化双布隆过滤器 BF_1 和 BF_2 的位数组 B_1 和 B_2；
2) 初始化 hostDict[0:m–1][0:4^k+2]←0；
3) 使用参数 HG、Pr、disData 和 k 并行构建敏感序列词典；
4) for s=0 to l–1 do //对 l 条长序列逐条计算
5) str[0:n_s–r][0:r–1]←0；
6) 依据 r 分割序列 N_s 为 n_s–r+1 条短序列，并将分割结果存入数组 str[0:n_s–r][0:r–1]；
7) **res**[0:n_s–1]←**0**；//并行初始化结果向量
8) for v=0 to Pr–1 para-do //CPU 并行计算
9) for d=0 to $\left\lfloor\frac{n_s-r+1}{\mathrm{Pr}}\right\rfloor$–1 do
10) 从 str[v+dPr][r–1]提取 $R_{s,v+d\mathrm{Pr}}$
11) 使用 BF_1 和 BF_2 对 $R_{s,v+d\mathrm{Pr}}$ 过滤去重
12) if $R_{s,v+d\mathrm{Pr}}$ 由 BF_1 过滤出 then
13) res[v+dPr]←1；
14) else if $R_{s,v+d\mathrm{Pr}}$ 由 BF_2 过滤出 then

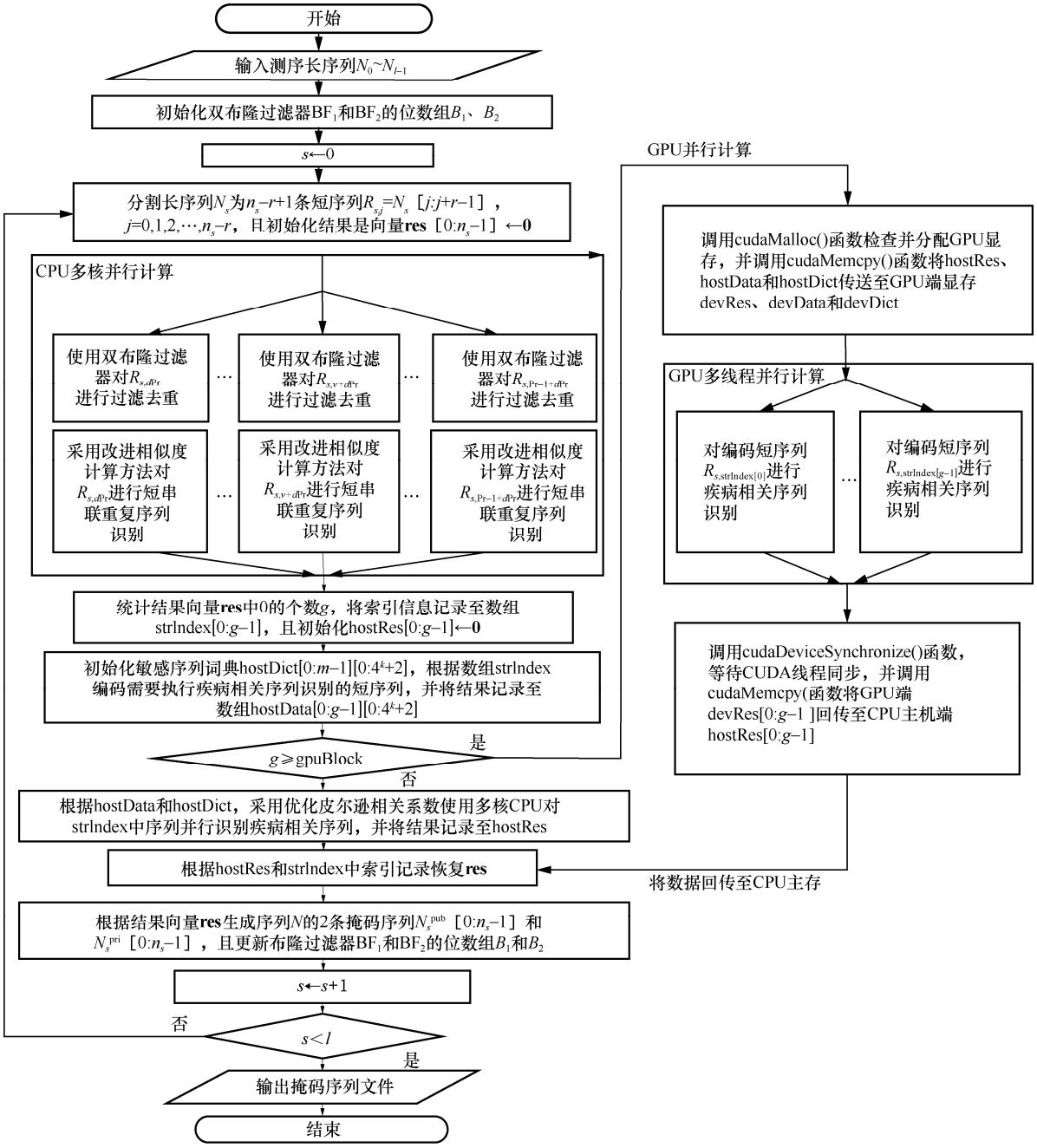

图 2　CPU/GPU 并行算法 CGPU-F3SR 的处理流程

15)　　res[v+dPr]←2;
16)　　else
17)　　依据 k、r 和式(3)计算 $R_{s,v+d\mathrm{Pr}}$ 的局部片段相似度 sim(X,Y);
18)　　if sim(X,Y)≥str_sim then
19)　　res[v+dPr]←3;
20)　　end if
21)　　end if
22)　　end for
23)　　end for
24)　　$g\leftarrow\sum_{j=0}^{n_s-r} I\left(\mathrm{res}[j]\right)=0$;
25)　　hostRes[0:$g-1$]←0，strIndex [0:$g-1$]←0，hostData[0:$g-1$][0:4^k+2]←0;
26)　　for j=0 to n_s-r do //构造辅助数组 strIndex
27)　　if res[j]=0 then
28)　　strIndex[i]←j;
29)　　i←i+1;

30) end if
31) end for
32) for v=0 to Pr–1 para-do //CPU 并行计算
33) for i=0 to $\left\lceil \frac{g}{\mathrm{Pr}} \right\rceil$–1 do
34) 对 str[strIndex[v+iPr]][0:r–1]进行 k-mer 编码，统计 G、C、A 碱基含量，且将计算得到的 k-mer 编码结果和 G、C、A 碱基含量结果存储至 hostData[v+iPr][0:4^k+2];
35) end for
36) end for
37) if g≥gpuBlock then //GPU 并行计算
38) CPU 调用 cudaMelloc()分配并检查 GPU 端 devData[0:g–1][0:4^k+2]、devRes [0:g–1]、devDict[0:m–1][0:4^k+2]显存空间;
39) CPU 调用 cudaMemcpy()将 hostRes、hostDict 和 hostData 传到 GPU 端 devRes、devDict 和 devData;
40) gpuGrid←$\left\lceil \frac{g}{\mathrm{gpuBlock}} \right\rceil$; //计算网格大小
41) tID←gpuBlock×blockIdx.x+threadIdx.x; //计算 CUDA 块内线程索引
42) for i=0 to m–1 do
43) 按式(4)计算 sim(D_i,$R_{s,\mathrm{tID}}$);
44) if sim(D_i,$R_{s,\mathrm{tID}}$)≥dis_sim then
45) devRes[tID]←4;
46) end if
47) end for
48) GPU 调用 cudaMemcpy()将 devRes 回传至 CPU 端 hostRes;
49) else
50) CPU 端依据 hostData、hostDict 并行识别疾病相关序列并将结果保存至 hostRes;
51) end if
52) 依据 strIndex 将 hostRes 结果恢复至 **res**;
53) 依据 **res** 和 str 更新布隆过滤器;
54) 依据 **res** 参考文献[15]的方法生成序列 N_s[0:n_s–1]的掩码敏感序列 $N_s^{\mathrm{Pri}}[0:n_s-1]$ 和掩码非敏感序列 $N_s^{\mathrm{Pub}}[0:n_s-1]$;
55) end for

设测序序列的平均长度为 n，在算法 1 中，步骤 1)～步骤 3)初始化布隆过滤器和敏感序列词典的时间复杂度为 $O\left(\frac{mr4^k}{\mathrm{Pr}}\right)$，步骤 4)～步骤 6)分割长序列的时间复杂度为 $O(nl)$，步骤 7)初始化结果向量 **res** 的时间复杂度为 $O\left(\frac{nl}{\mathrm{Pr}}\right)$，步骤 8)～步骤 23)并行短序列过滤去重和短串联重复序列识别的时间复杂度为 $O\left(\frac{nl(h_1+h_2)}{\mathrm{Pr}}+\frac{nrl4^k}{\mathrm{Pr}}\right)$，步骤 24)～步骤 52)并行识别疾病相关序列的时间复杂度为 $O\left(\frac{nml4^k}{\mathrm{Cu}}\right)$，步骤 53)更新布隆过滤器时间复杂度为 $O(nl)$。步骤 54)和步骤 55)生成掩码核苷酸序列的时间复杂度为 $O(lnr^2)$。由于 h_1、h_2、k 和 r 均为常数，因此并行算法 CGPU-F3SR 的时间复杂度为 $O\left(\frac{mr4^k}{\mathrm{Pr}}+nl+\frac{nl}{\mathrm{Pr}}+\frac{nl(r4^k+h_1+h_2)}{\mathrm{Pr}}+\quad lnr^2+\frac{nml4^k}{\mathrm{Cu}}\right)=O\left(\frac{nl}{\mathrm{Pr}}+\frac{nml}{\mathrm{Cu}}\right)=O\left(\frac{nl(m\mathrm{Pr}+\mathrm{Cu})}{\mathrm{PrCu}}\right)$。

在算法 CGPU-F3SR 运行过程中，数组 N、N_s^{pub} 和 N_s^{Pri} 所需的存储空间为 $O(n)$。布隆过滤器位数组 B_1 和 B_2 所需的存储空间为 $O(|b_1+b_2|)=O=\left(\left|\frac{t_1\ln\left(p_1^{-1}\right)+t_2\ln\left(p_2^{-1}\right)}{(\ln 2)^2}\right|\right)$。敏感序列词典 hostDict 和 devDict 二维向量所需的存储空间为 $O(m4^k)$。算法使用 Pr 个 CPU 核心编码核苷酸序列所需的存储空间为 $O(r\mathrm{Pr}4^k)$。结果向量 **res** 所需的存储空间为 $O(n)$。数组 str 所需的存储空间为 $O(n4^k)$。二维数组 hostData 和 devData 所需的存储空间为 $O(n4^k)$。数组 hostRes 和 devRes 所需的存储空间为 $O(n)$。对于序列分割得到的短序列始终不存在重复短序列的情形（需要存储空间最多的情形），所有短序列的敏感识别结果均要更新至双布隆过滤器位数组，此时双布隆过滤器可插入的最大序列数目满足 $t_i=a_inl$，a_i≥1，i=1,2。由于 k 和 a_i 均为常数，n>m，i=1,2，因此算法 CGPU-F3SR 在 CPU 端所需的最大存储空间为 $O\left(\left\{\max\left|\frac{t_1\ln\left(p_1^{-1}\right)+t_2\ln\left(p_2^{-1}\right)}{(\ln 2)^2}\right|,n,(m+n+r\mathrm{Pr})4^k\right\}\right)=O((a_1+a_2)nl)=O(nl)$，在 GPU 端所需的显存空间为 $O((m+n)4^k)=O(n)$。因此，并行算法 CGPU-F3SR 的空间复杂度为 $O(nl)$。

当 CPU/GPU 并行算法 CGPU-F3SR 中参数的 gpuBlock 取值为 0 时，它就变为仅采用多核 CPU 并行计算的版本（简记为算法 CPU-F3SR）。

2　实验

实验在广西大学的 Sugon 7000A 超级并行计算机上进行，使用 CPU/GPU 异构计算节点。每个计算节点的内存容量为 512 GB、外部存储容量为 8×900 GB、CPU 为 2×Intel Xeon Gold 6230（总内核数为 40）、GPU 为 NVIDIA Tesla-T4（显存容量为 16 GB，含有 2 560 个 CUDA 核心）。运行操作系统为 64 位版本 CentOS7.4。采用 C++、CUDA 和 OpenMP 混合编程实现并行算法 CGPU-F3SR。

实验采用 GWAS Catalog 数据库[18]中全基因组关联研究数据集 gwas_catalog_v1.0.tsv 构建敏感序列词典，该数据集包含 Scoliosis、HIV-1 replication 和 Opioid sensitivity 等 4 680 种疾病，共计 216 250 组数据。采用的参考基因组为 NCBI 数据库[19]开放的人类基因组 HG38。实验所用的疾病易感基因来自 GWAS Catalog 数据库[18]，短串联重复序列来自 TRDB 数据库[20]。参考文献[3]的数据预处理规则，预处理每条序列长度为 50 bp 的基准数据序列集 All-Together（2 239 340 条序列），每个基准数据集的非敏感序列和敏感序列数量之比为 7:1。

为评估算法识别效果，需获取不同长度且带有敏感序列标签的测序长序列。为此，实验对错误率为 2%～20%的 All-Together 基准数序列集进行随机抽样，生成序列长度为 100～400 kbp 的总规模约 24 GB 的 4 组长序列数据集 DataSet-100 kbp～DataSet-400 kbp 用于并行算法参数调优测试。

实验将本文 CPU 和 GPU 协同计算的并行算法 CGPU-F3SR 与并行算法 PARA-LRF[11]、仅采用 CPU 并行计算的版本 CPU-F3SR（gpuBlock=0）、串行算法 F3SR[15]进行测试比较识别效果。实验评估算法的识别准确率（acc, accuracy）[3]、查准率（pre, precision）[4]和假阳性率（fpr, false positive rate）[4]。准确率和查准率越高，算法识别效果越好；假阳性率越低，算法识别效果越好。在运行效率方面，实验评估了算法的识别吞吐量（thp, throughput）[4]。吞吐量定义为算法单位时间识别脱氧核糖核苷酸的数量，单位为 nt/s[3-4]，吞吐量越高，算法运行效率越高。

下面，首先给出本文所提出的 CPU/GPU 并行算法 CGPU-F3SR 的参数调优实验结果，然后给出消融实验结果，最后给出算法 CGPU-F3SR 和同类并行算法的实验结果及分析。

2.1　并行算法参数调优实验

参考文献[3,4,15]给出的基因组数据敏感序列识别算法研究的相关实验，算法 CGPU-F3SR 选取 p_i=0.000 1、t_i=10^8、r=48 bp、k=5、str_sim=0.64、dis_sim=0.80 进行算法部分参数的初始化，i=1,2。

为确定 GPU 线程块大小对并行算法吞吐量的影响，实验测试了并行算法 CGPU-F3SR 采用不同的 GPU 线程块大小在 4 组长序列数据集上运行时的吞吐量，结果如图 3 所示。

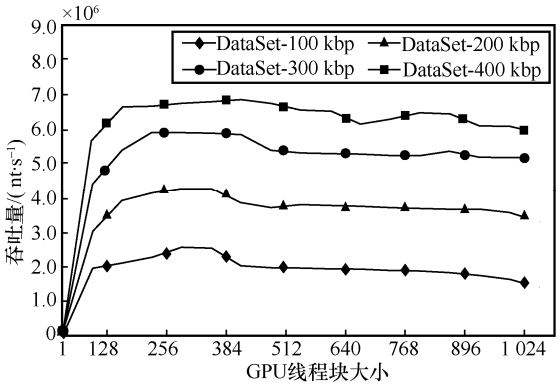

图 3　并行算法 CGPU-F3SR 采用不同的 GPU 线程块大小在 4 组长序列数据集上运行时的吞吐量

图 3 实验结果表明，当 GPU 线程块大小固定时，序列数据集规模越大，并行算法 CGPU-F3SR 的吞吐量越高。原因如下：1) 测序长序列数据集规模越大，CGPU-F3SR 分割长序列得到的短序列中的重复短序列越多，由于双布隆过滤避免了重复短序列的多次运算，因此过滤效果明显；2) CGPU-F3SR 将编码后的短序列提交给 GPU 进行疾病相关序列并行识别，测序数据集越大，GPU 线程平均计算处理的数据越多，算法的并行度越高。按照 CUDA 并行模型，网格 Grid 中的各线程块 Block 会被分配到 GPU 的各流式处理器中执行。图 3 的实验结果还表明，当参数 gpuBlock 取值为 128～384 时，算法整体上具有较高的吞吐量。

图 4 进一步给出了 GPU 线程块大小 gpuBlock=256、使用不同 CPU 核心数运行并行算法 CGPU-F3SR 时得到的吞吐量结果。

图 4 实验结果表明，并行算法 CGPU-F3SR 的识别吞吐量与使用的 CPU 核心数呈正相关。数据集规模越大，序列越长，算法吞吐量受 CPU 核心数增益效果越明显。

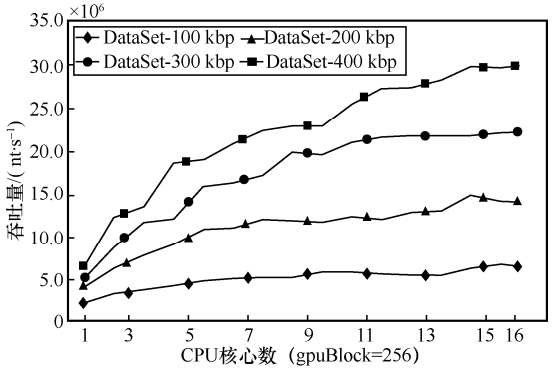

图 4 并行算法 CGPU-F3SR 使用不同 CPU 核心数在 4 组长序列数据集上运行时的吞吐量

为使本文并行算法 CGPU-F3SR 充分发挥 CPU 和 GPU 协同并行计算的能力，根据图 3 和图 4 的实验结果，本文选取 gpuBlock=256 和 Pr=4 的组合进行后续的实验。

2.2 并行算法消融实验

并行算法 CGPU-F3SR 引入布隆过滤器对序列数据进行过滤，以免对分割短序列的重复计算。为了确定过滤机制对并行算法 CGPU-F3SR 的性能影响，本文实验评估了并行算法 CGPU-F3SR 进行序列过滤 With Filter（参数 Pr=4，gpuBlock=256）和不进行序列过滤 Without Filter（参数 Pr=30，gpuBlock=256）2 种情形在 4 组长序列数据集运行得到的识别准确率、查准率、假阳性率和吞吐量，实验结果如表 1 所示。

表 1 的消融实验结果表明，在识别性能方面，并行算法 CGPU-F3SR 在 With Filter 和 Without Filter 这 2 种情形下均获得相同的识别准确率、查准率和假阳性率。这是因为：1) 布隆过滤器的多哈希映射机制可以对相同序列产生多个相同的哈希映射，这可以在避免对重复序列的多次计算的同时不影响其识别性能；2) 改进序列相似度计算模型能对含有错误信息的相似序列产生近似度量值，并行算法 CGPU-F3SR 的识别准确率、查准率和假阳性率均取决于改进的序列相似度计算模型，而不是取决于布隆过滤器。在运行效率方面，算法使用 4 个 CPU 核心的 With Filter 模式的识别吞吐量比使用 30 个 CPU 核心的 Without Filter 模式高了 2 个数量级。这是因为长序列基因组数据中分割短序列具有大量相同的重复短序列，布隆过滤效果明显。

2.3 并行算法识别性能实验

为评估算法的并行化对高错误率长序列基因数据中敏感序列（包括 STR 和 DRS）识别效果的影响，首先，实验测试 CPU 和 GPU 协同并行算法 CGPU-F3SR、仅 CPU 并行的算法 CPU-F3SR 和串行算法 F3SR（参数取值 p_i=0.000 1，t_i=10^4，r=48，k=5，dis_sim=0.80，str_sim = 0.64，i=1,2）[15]在 4 组长序列数据集上运行得到的识别准确率、查准率和假阳性率，实验结果如表 2 所示。

表 2 的实验结果表明，CPU 与 GPU 协同并行算法 CGPU-F3SR 和 CPU 并行化算法 CPU-F3SR 均获得了与串行算法 F3SR 相同的识别结果。这是因

表 1 并行算法 CGPU-F3SR 的识别准确率、查准率、假阳性率和吞吐量

长序列数据集	CGPU-F3SR (With Filter)				CGPU-F3SR (Without Filter)			
	acc	pre	fpr	thp/(nt·s^{-1})	acc	pre	fpr	thp/(nt·s^{-1})
DataSet-100 kbp	94.827%	91.333%	0.757%	4.280×10^6	94.827%	91.333%	0.757%	0.368×10^6
DataSet-200 kbp	94.829%	91.327%	0.758%	9.162×10^6	94.829%	91.327%	0.758%	0.402×10^6
DataSet-300 kbp	94.827%	91.323%	0.758%	12.109×10^6	94.827%	91.323%	0.758%	0.430×10^6
DataSet-400 kbp	94.830%	91.320%	0.759%	18.581×10^6	94.830%	91.320%	0.759%	0.423×10^6
平均值	94.828%	91.326%	0.758%	11.033×10^6	94.828%	91.326%	0.758%	0.406×10^6

表 2 并行算法 CGPU-F3SR、CPU-F3SR 和 F3SR 的识别准确率、查准率和假阳性率

算法	CGPU-F3SR			CPU-F3SR			F3SR		
长序列数据集	acc	pre	fpr	acc	pre	fpr	acc	pre	fpr
DataSet-100 kbp	94.827%	91.333%	0.757%	94.827%	91.333%	0.757%	94.827%	91.333%	0.757%
DataSet-200 kbp	94.829%	91.327%	0.758%	94.829%	91.327%	0.758%	94.829%	91.327%	0.758%
DataSet-300 kbp	94.827%	91.323%	0.758%	94.827%	91.323%	0.758%	94.827%	91.323%	0.758%
DataSet-400 kbp	94.830%	91.320%	0.759%	94.830%	91.320%	0.759%	94.830%	91.320%	0.759%
平均值	94.828%	91.326%	0.758%	94.828%	91.326%	0.758%	94.828%	91.326%	0.758%

为长序列分割得到的相邻短序列之间共享 r–1 个高度重叠的碱基，通过 CPU/GPU 并行计算不会影响算法的最终识别质量。在上述实验的基础上，进一步测试比较本文并行算法 CGPU-F3SR 和同类并行算法 PARA-LRF（参数 p=0.02，m=68 915 861）[11] 在 DataSet-100 kbp～DataSet-400 kbp 数据集上运行时，对高错误率长序列基因组数据中敏感序列（包括 STR 和 DRS）进行识别获得的识别准确率、查准率和假阳性率，实验结果如表 3 所示。

表 3 的实验结果表明，相较于并行算法 PARA-LRF，本文 CPU/GPU 并行算法 CGPU-F3SR 整体上具有更高的识别准确率和查准率以及更低的识别假阳性率。这是因为并行算法 CGPU-F3SR 采用 k-mer 编码策略有效地提取了高错误率长序列中的碱基错误信息，且根据 STR 和 DRS 的结构特点，通过引入控制参数改进了 STR 和 DRS 相似度的计算方法，使本文并行算法 CGPU-F3SR 在准确识别出高错误率长序列基因组数据中敏感序列的同时，降低了将非敏感序列误判为敏感序列的情形的发生，从而整体上获得更好的识别效果。

2.4　并行算法运行效率实验

在基因组数据敏感序列识别研究中，算法吞吐量和运行时间相关，多用于运行效率评估。基因组数据敏感序列识别并行算法相关研究较少，目前缺乏 CPU 和 GPU 协同计算的并行算法。并行算法 PARA-LRF[11]是基于多核 CPU 设计的并行算法。为公平起见，实验首先测试了本文仅多核 CPU 并行算法 CPU-F3SR 和多核 CPU 并行算法 PARA-LRF [11]使用不同 CPU 核心数在 4 组长序列数据集上运行得到的识别吞吐量，结果如图 5 所示。

表 3　　并行算法 CGPU-F3SR 和 PARA-LRF 的识别准确率、查准率和假阳性率

并行算法	PARA-LRF			CGPU-F3SR		
长序列数据集	acc	pre	fpr	acc	pre	fpr
DataSet-100 kbp	87.066%	48.310%	8.161%	94.827%	91.333%	0.757%
DataSet-200 kbp	87.055%	48.280%	8.171%	94.829%	91.327%	0.758%
DataSet-300 kbp	87.059%	48.288%	8.170%	94.827%	91.323%	0.758%
DataSet-400 kbp	87.059%	48.287%	8.167%	94.830%	91.320%	0.759%
平均值	87.060%	48.291%	8.167%	94.828%	91.326%	0.758%

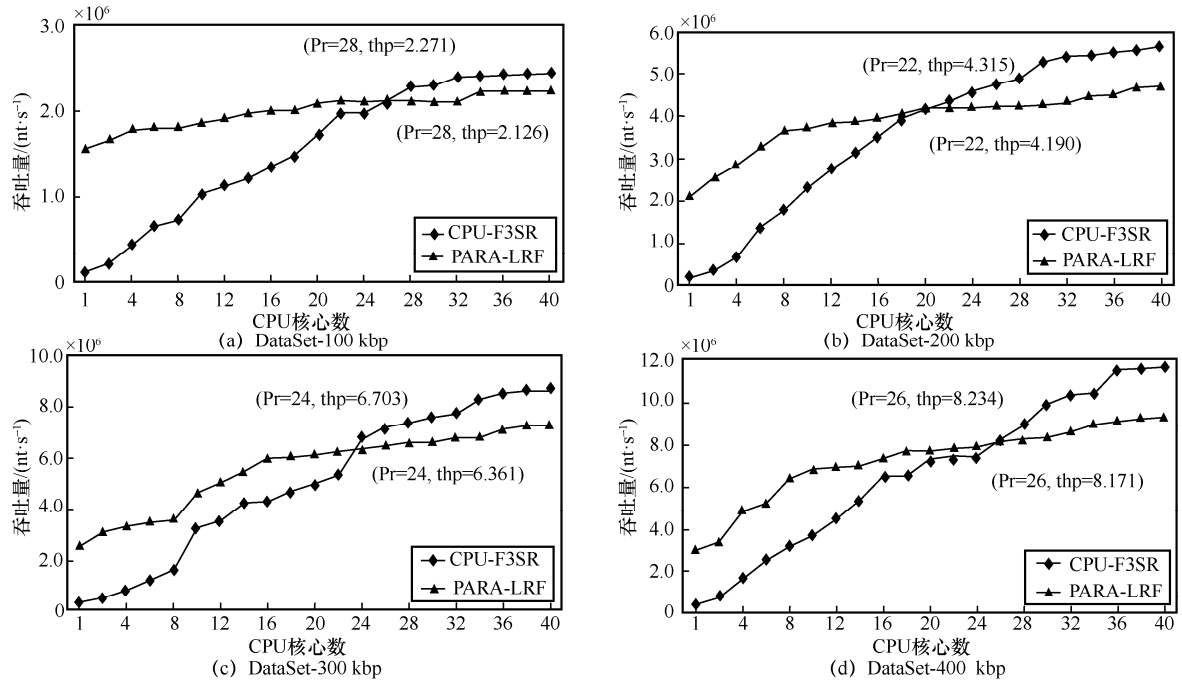

图 5　多核 CPU 并行算法 CPU-F3SR 和 PARA-LRF 使用的 CPU 核心数对吞吐量的影响

从图 5(a)可以看出，在 DataSet-100 kbp 数据集上，当使用的 CPU 核心数为 1～28 时，CPU 并行算法 PARA-LRF 的识别吞吐量高于本文 CPU 并行算法 CPU-F3SR；当使用的 CPU 核心数为 28～40 时，并行算法 CPU-F3SR 比 PARA-LRF 获得了整体上更高的识别吞吐量。

从图 5(b)～图 5(d)可以看出，对序列长度为 200～400 kbp 的 3 个长序列数据集，当 CPU 核心数分别为 22、24 和 26 时，并行算法 CPU-F3SR 达到吞吐量拐点，此时本文多核 CPU 并行算法 CPU-F3SR 比多核 CPU 并行算法 PARA-LRF 整体上具有更高的识别吞吐量。这是因为并行算法 CPU-F3SR 并行过滤去重分割得到的短序列、并行识别短串联重复序列、并行识别疾病相关序列的部分占比高、算法并行度高，所以在 4 组长序列数据集上运行时获得了更高的识别吞吐量。

从长序列基因组数据中识别疾病相关序列的过程是一个计算密集型工作，为进一步加速求解敏感序列识别问题，本文设计实现了 CPU 和 GPU 协同计算的敏感序列识别并行算法 CGPU-F3SR。为此，实验也测试了 CPU/GPU 并行算法 CGPU-F3SR（参数 Pr=4、gpuBlock=256）、多核 CPU 并行算法 PARA-LRF（参数 p=0.02、m=68 915 861、Pr=4）[11]、多核 CPU 并行算法 CPU-F3SR（参数 Pr=4）在 4 组长序列数据集上运行得到的识别吞吐量，结果如表 4 所示。

从表 4 可以看出，相较于本文仅采用多核 CPU 并行加速的 CPU-F3SR 算法，本文的 CPU 和 GPU 协同并行求解敏感序列识别问题的 CGPU-F3SR 算法平均吞吐量高达 11.033×10^6 nt/s，极大提升了识别吞吐量；相较于已有的多核 CPU 并行算法 PARA-LRF，本文算法 CGPU-F3SR 在吞吐量方面也具有较大幅度的提升。CGPU-F3SR 算法的平均吞吐量分别是 CPU-F3SR 和 PARA-LRF 算法的 11.45 倍和 2.44 倍。CGPU-F3SR 算法之所以获得更高的吞吐量，是因为它采用 GPU 并行加速处理属于计算密集型的疾病相关序列识别过程，加速效果显著。

综上所述，CPU 和 GPU 协同计算的并行算法 CGPU-F3SR 在准确识别出大规模高错误率长序列基因组数据中的 2 类敏感序列（STR 和 DRS）的同时，显著提升了吞吐量。

3 结束语

第三代测序技术产生的基因组测序数据正以超摩尔定律的速度显著增长。随着人们生物信息安全意识的逐渐增强，保护个体基因组数据敏感序列变得越来越重要。本文提出的 CPU 和 GPU 协同计算识别的并行算法 CGPU-F3SR 的特色和优势是高效准确地识别出第三代测序技术产生的大规模的高错误率长序列基因组数据中的 2 类敏感序列（短串联重复序列和疾病相关序列）。在长度为 100～400 kbp 的长序列数据集上的实验结果表明，并行算法 CGPU-F3SR 在整体上获得了较高的识别准确率、查准率和较低假阳性率的同时，显著提升了识别吞吐量。

近些年来，学者研究发现基因组测序数据中还存在“均聚物”错误。考虑到基因组数据错误类型复杂、数据规模巨大、数据敏感的特点，在本文研究工作基础上，下一步的研究方向将包括：1) 研究设计能准确识别出“均聚物”错误的敏感序列识别算法；2) 探索设计新的数据索引结构或借鉴其他索引数据结构（如简洁 de Bruign 图结构、FM-index 结构等），研究设计在 CPU/GPU 混合体系结构集群上实现的并行算法，以进一步加速求解超大规模超长序列的基因组数据敏感序列识别问题；3)研究如何将本文提出的敏感序列识别并行算法与安全比对算法进行级联，以实现大规模的高错误率长序列基因组数据的安全比对。

表 4　并行算法 CGPU-F3SR、CPU-F3SR 和 PARA-LRF 的识别吞吐量

长序列数据集	CGPU-F3SR	CPU-F3SR	PARA-LRF
DataSet-100 kbp	4.280×10^6	0.310×10^6	2.112×10^6
DataSet-200 kbp	9.162×10^6	0.691×10^6	3.844×10^6
DataSet-300 kbp	12.109×10^6	0.857×10^6	5.133×10^6
DataSet-400 kbp	18.581×10^6	1.997×10^6	6.990×10^6
平均值	11.033×10^6	0.964×10^6	4.519×10^6

参考文献：

[1] LU D D, ZHANG Y, ZHANG L, et al. Methods of privacy-preserving genomic sequencing data alignments[J]. Briefings in Bioinformatics, 2021, 22(6): 1-15.

[2] 刘海, 彭长根, 吴振强, 等. 基因组数据隐私保护理论与方法综述[J]. 计算机学报, 2021, 44(7): 1430-1480.
LIU H, PENG C G, WU Z Q, et al. A survey of the theories and methods of privacy preserving of genome data[J]. Chinese Journal of Computers, 2021, 44(7): 1430-1480.

[3] COGO V V, BESSANI A, COUTO F M, et al. A high-throughput method to detect privacy-sensitive human genomic data[C]// Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society. New York: ACM Press, 2015: 101-110.

[4] DECOUCHANT J, FERNANDES M, VÖLP M, et al. Accurate filtering of privacy-sensitive information in raw genomic data[J]. Journal of Biomedical Informatics, 2018, 82: 1-12.

[5] LOKA T P, TAUSCH S H, DABROWSKI P W, et al. PriLive: privacy-preserving real-time filtering for next-generation sequencing[J]. Bioinformatics (Oxford, England), 2018, 34(14): 2376-2383.

[6] CHEN J X, LI F Y, WANG M, et al. BigFiRSt: a software program using big data technique for mining simple sequence repeats from large-scale sequencing data[J]. Frontiers in Big Data, 2022, 4: 727216.

[7] 陆向艳, 钟诚. 机群系统上长序列最大串联重复识别并行算法[J]. 微电子学与计算机, 2010, 27(8): 186-189.
LU X Y, ZHONG C. Parallel algorithm for long sequences maximal tandem repeats on the cluster computing systems[J]. Microelectronics & Computer, 2010, 27(8): 186-189.

[8] GONZÁLEZ-DOMÍNGUEZ J, MARTÍN-MARTÍNEZ J M, EXPÓSITO R R. MPI-dot2dot: a parallel tool to find DNA tandem repeats on multicore clusters[J]. The Journal of Supercomputing, 2022, 78(3): 4217-4235.

[9] HUANG C H, RAJASEKARAN S. Parallel pattern identification in biological sequences on clusters[C]//Proceedings of IEEE Transactions on NanoBioscience. Piscataway: IEEE Press, 2002: 29-34.

[10] NAKAMURA A, TAKIGAWA I, MAMITSUKA H. Efficiently enumerating substrings with statistically significant frequencies of locally optimal occurrences in gigantic string[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 34(4): 5240-5247.

[11] FERNANDES M. Reconciling data privacy with sharing in next-generation genomic workflows[D]. Luxembourg: University of Luxembourg, 2020.

[12] LOGSDON G A, VOLLGER M R, EICHLER E E. Long-read human genome sequencing and its applications[J]. Nature Reviews Genetics, 2020, 21(10): 597-614.

[13] 万盛, 何媛媛, 李凤华, 等. 基于布隆过滤器的轻量级隐私信息匹配方案[J]. 通信学报, 2015, 36(12): 151-162.
WAN S, HE Y Y, LI F H, et al. Bloom filter-based lightweight private matching scheme[J]. Journal on Communications, 2015, 36(12): 151-162.

[14] MORISHITA S, ICHIKAWA K, MYERS E W. Finding long tandem repeats in long noisy reads[J]. Bioinformatics, 2020, 37(5): 612-621.

[15] 孙辉, 钟诚. 融合过滤和相似度计算的高错误率基因组数据敏感序列识别[J]. 小型微型计算机系统, 2022: doi.org:/10.20009/j.cnki.21110 6/TP.2021-0766.
SUN H, ZHONG C. Recognizing sensitive sequences from genomic data with high error rate integrating filter and similarity calculation[J]. Journal of Chinese Computer Systems, 2022: doi.org:/10.20009/j.cnki.21-110 6/TP.20 21-0766.

[16] LI H, HANDSAKER B, WYSOKER A, et al. The sequence alignment/map format and SAMtools[J]. Bioinformatics, 2009, 25(16): 2078-2079.

[17] NICHOLAS W. CUDA 专家手册: GPU 编程权威指南[M]. 苏统华, 马培军, 译. 北京: 机械工业出版社, 2014.
NICHOLAS W. CUDA expert manual: the definitive guide to GPU programming[M]. Translated by SU T H, MA P J. Beijing: China Machine Press, 2014.

[18] BUNIELLO A, MACARTHUR J A L, CEREZO M, et al. The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019[J]. Nucleic Acids Research, 2018, 47(D1): 1005-1012.

[19] GEER L Y, MARCHLER-BAUER A, GEER R C, et al. The NCBI biosystems database[J]. Nucleic Acids Research, 2010, 38(D1): 492-496.

[20] GELFAND Y, RODRIGUEZ A, BENSON G. TRDB: the tandem repeats database[J]. Nucleic Acids Research, 2007, 35(D1): 80-87.

[作者简介]

钟诚（1964– ），男，广西桂平人，博士，广西大学教授、博士生导师，主要研究方向为并行分布计算、生物信息计算、网络信息安全。

孙辉（1997– ），男，宁夏银川人，广西大学硕士生，主要研究方向为并行计算、计算生物学、生物信息安全。