基因组大数据变异检测算法的并行优化

崔英博[1]，黄春[1]，唐滔[1]，杨灿群[1]，廖湘科[1]，彭绍亮[2,3]
1. 国防科技大学计算机学院，湖南 长沙 410073；2. 湖南大学信息科学与工程学院，湖南 长沙 410082；
3. 国家超级计算长沙中心，湖南 长沙 410082

摘要
序列比对和变异检测是基因组数据分析的基础步骤，是后续各种功能性分析的前提，也是基因组数据分析中最耗时的环节。为有效处理高通量测序技术产生的海量基因组大数据，采用OpenMP、MPI等技术，对序列比对算法和SNP检测算法进行了多级并行优化，并对相关算法进行了改进。在不同数据集和并行规模下的测试中，核心算法加速比达到9倍以上，大规模测试中算法的并行效率保持在60%以上，在保证精度的前提下获得了良好的并行性能和可扩展性，有效提高了基因组大数据变异检测的能力。

关键词
序列比对；SNP；OpenMP；MPI

中图分类号：TP391 文献标识码：A doi: 10.11959/j.issn.2096-0271.2020041

Parallel optimization of variation detection algorithms for large-scale genome data

CUI Yingbo[1], HUANG Chun[1], TANG Tao[1], YANG Canqun[1], LIAO Xiangke[1], PENG Shaoliang[2]
1. College of Computer, National University of Defense Technology, Changsha 410073, China
2. College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
3. National Supercomputer Center in Changsha, Changsha 410082, China

Abstract
Sequence alignment and mutation detection are the basic steps of genomic data analysis. They are the premise of subsequent functional analysis, and the most time-consuming steps. In order to effectively deal with the massive genomic big data brought by high-throughput sequencing technology, MPI, OpenMP and other technologies to perform multi-level parallel optimization of sequence alignment algorithm and SNP detection algorithm were used. By testing on different data sets and parallel scales, the core algorithm reached more than 9x speedup, and the parallel efficiency remained above 60% in large-scale test. The improved algorithms obtain good parallel performance and scalability, that effectively improves the ability of genomic big data mutation detection.

Key words
sequence alignment, SNP, OpenMP, MPI

1 引言

近年来，高通量测序技术的迅猛发展给生命科学带来了巨大变革，测序通量不断提高，测序成本不断下降，如单个人类基因组的测序成本逐年降低，基因组数据规模不断增长。基因组数据大约每7个月就会增加一倍[1]。现有服务器以及序列分析算法已经无法及时有效地处理如此大规模的数据，需要借助并行计算机的强大算力以及并行算法的有效支撑来实现对基因组大数据的有效处理[2]。

测序仪产生读段（reads）后，首先进行质量控制，去除测序质量较差的数据；然后进行序列比对，将读段回帖到基因组，找到读段最可能的起源位置，并输出比对文件；再基于比对文件检测基因组的变异情况，主要包括单核苷酸多态性（single nucleotide polymorphism，SNP）、插入删除变异（indel）、结构变异（structure variation，SV）和拷贝数变异（copy number variation，CNV）等；最后根据需要，进行特定的功能分析。序列比对和变异检测是基因组数据分析的基础环节，是后续功能性分析的基础，也是数据分析流程中最耗时的步骤[3]。

为有效处理高通量测序技术带来的海量基因组大数据，本文选取基因组变异检测中常用的序列比对和SNP检测两个步骤，对相关算法进行了改进，并利用OpenMP（open multi-processing）和消息传递接口（message passing interface，MPI）等并行技术实现了多级并行。在不同数据集和并行规模下的测试中，核心算法的加速比达到9倍以上，大规模测试中算法的并行效率保持在60%以上，在保证精度的前提下获得了良好的并行性能和可扩展性，有效提高了基因组大数据变异检测的能力。

2 相关工作

2.1 序列比对算法

序列比对算法以读段和参考基因组为输入，将读段比对到基因组，找到读段最可能的起源位置。基因组规模一般较大，且高通量测序产生的读段数量庞大，现有的比对工具大部分是借助索引结构来处理大规模数据的。根据索引结构的不同，可以将序列比对算法分为两类：基于哈希表的比对算法和基于后缀树的比对算法[4]。

最早的基于哈希索引的比对工具是BLAST[5]，后续出现的基于哈希索引的比对工具有SOAP[6]、SeqMap[7]、RMAP[8]、BFAST[9]、CloudBurst[10]、PerM[11]和GNUMAP[12]等。

后缀树是一种存储着一个字符串所有后缀的数据结构，在后缀树中，所有相同的后缀都会合并成一条路径。因此，基于后缀树的序列比对可以合并对相同后缀的比较，提高效率。基于后缀树的比对算法查询时间为$O(q)$，其中q为子串的长度，时间复杂度与母串无关，具有较高的查询效率。但是，后缀树的空间复杂度较高，内存需求很大，无法在常用的服务器上实现索引构建，因此基于后缀树的比对算法在早期应用较少。

FM索引[13]的出现大大降低了后缀树的空间开销，使得基于后缀树的比对算法得到了广泛应用。FM索引可以被理解为压缩的后缀树，其能够在压缩状态下实现字符串搜索，因此极大地降低了索引的空间开销，如人类基因组的FM索引大小为3 GB左右，大多数服务器甚至台式机能满足需

求。基于FM索引的比对算法具有时间和空间的双重优势，迅速在二代序列比对领域流行起来。常用的基于FM索引的比对工具有BWA[14]、Bowtie[15]和SOAP2[16]等，本文的工作基于BWA开展。

2.2 SNP检测算法

SNP是基因组中单个碱基位点的变异。序列比对完成后，基因组中大多数位点会对应多条读段，SNP检测通过逐个位点比较参考基因组和输入序列来确定SNP位点。每个位点的SNP检测过程完全相同，且相互独立。SNP检测算法一般会综合位点的碱基类型、测序质量、在基因组中的位置等信息，计算当前位点是SNP位点的概率。基因组规模一般较大，而内存空间有限，无法一次性完成全基因组的SNP检测，因此SNP检测算法一般会将基因组分为若干窗口，对窗口进行逐个检测。

SNP检测算法将比对完成的读段、参考序列作为输入，有时也将已知SNP的数据库信息作为输入[17]。研究人员开发了一些基于Web的SNP检测工具，只要通过浏览器上传相关数据，即可完成分析，包括HaploSNPer[18]和SNiPlay[19]等。但是基于Web的分析工具在数据安全性方面存在一定的风险，因此，更常用的工具是能够独立运行的SNP检测工具，包括QualitySNP[20]、SAMtools[21]、SOAPsnp[17]、GATK[22]、Isaac[23]、SNVer[24]、VarScan[25]和VarScan2[26]等，本文的工作基于SOAPsnp开展。

SNP分析是一个相对耗时的过程，因此，也存在不少对SNP算法的优化，如Crossbow[27]利用Hadoop和云计算加速SNP检测，Rainbow[28]针对大数据集对Crossbow做了进一步优化，GSNP[29]通过GPU来加速SNP的计算过程。

2.3 OpenMP和MPI技术

如**图1**、**图2**所示，从硬件实现的角度，可以将并行计算机系统分为共享内存系统和分布式内存系统两种。在共享内存系统中，多个计算核心或CPU共享内存，可以提高内存数据的利用率；在分布式内存系统中，每个节点拥有自己独立的CPU和内存，节点间通过互联网络连接，CPU只能访问自己节点上的内存空间，无法访问其他节点上的内存空间。

针对两种不同的并行系统，存在两种不同的并行编程模型。面向共享内存系统的并行编程模型以多线程为主，主要包括OpenMP和pthread，其中pthread需要程序员控制多线程的发起、数据分配、同步等操作，较为烦琐。而OpenMP仅需要在程序中可并行的模块前添加指导语句即可，由编译器和运行时库负责对数据进行划分、发起多线程并行、管理私有和共享数据、进行线程同步等。相比之下，OpenMP更易于实现，而且具有良好的可移植性[30-31]，因此本文主要基于OpenMP实现多线程并行。

面向分布式内存系统的并行编程模型以MPI为主，支持多进程并行，进程间通过传递消息的方式实现跨节点通信。MPI并不是一种新的编程语言，而是一个通信库，支持C、C++和Fortran等语言，可以看作对这些语言的扩展。MPI除了支持点对点通信外，还支持广播和规约等聚合通信，非常便利，是分布式系统并行编程的事实标准。

3 相关算法分析

3.1 序列比对算法分析

序列比对算法一般包括读入读段、比

对和写出结果3个步骤。为提高比对效率，一般会事先对参考基因组建立索引。对于给定物种，参考基因组的索引生成后可以被重复使用，对整体比对时间的影响并不大，因此，本文主要对比对过程进行并行优化。

BWA是使用非常广泛的、基于Burrows-Wheeler变换（Burrows-Wheeler transform，BWT）的短序列比对工具之一[32]。BWT是一种数据压缩算法，其主要思想是对一个给定文本的所有轮转进行排序，返回的排序的最后一列被称为BWT字符串，该列为BWT的结果。文本经过BWT后会将很多相同的字符排序到一起，因此更加容易进行压缩。FM索引是一种基于BWT的索引结构，支持在压缩状态下对文本进行字符串查询。在BWA中，精确匹配采用后向搜索方法实现，即对于一个字符串，从后向前逐个字符地进行比对，后向搜索实际上完成了对基因组构成的后缀树的自顶向下遍历。

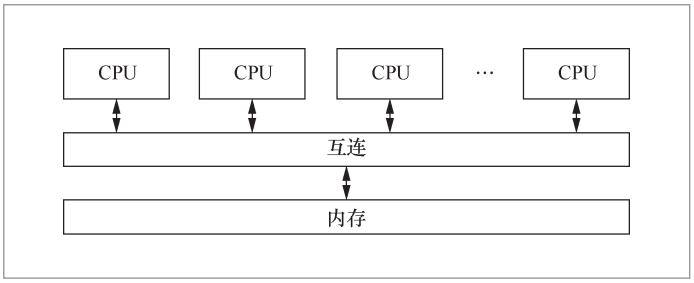


图1 共享内存系统

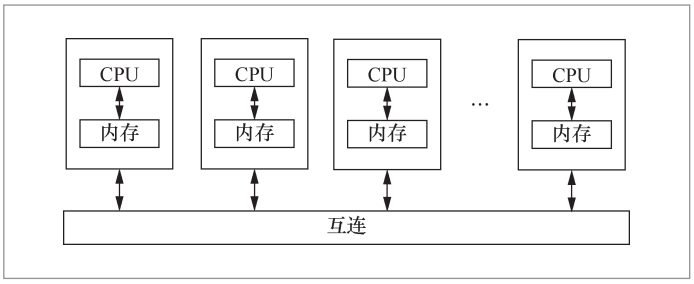


图2 分布式内存系统

3.2 SNP检测算法分析

图3为SOAPsnp检测SNP的流程。SOAPsnp的输入包括比对的读段和参考基因组，有时也包括已知SNP的数据库信息，SOAPsnp的输出为保守基因型信息。SOAPsnp主要包括7个模块：cal_p_mat、read_site、counting、likelihood、posterior、output和recycle，核心数据结构包括p_matrix、base_info和type_likely。

cal_p_mat模块扫描读段比对文件，计算满足四元组合(碱基质量分数，读段中的偏移，参考基因组碱基类型，读段中

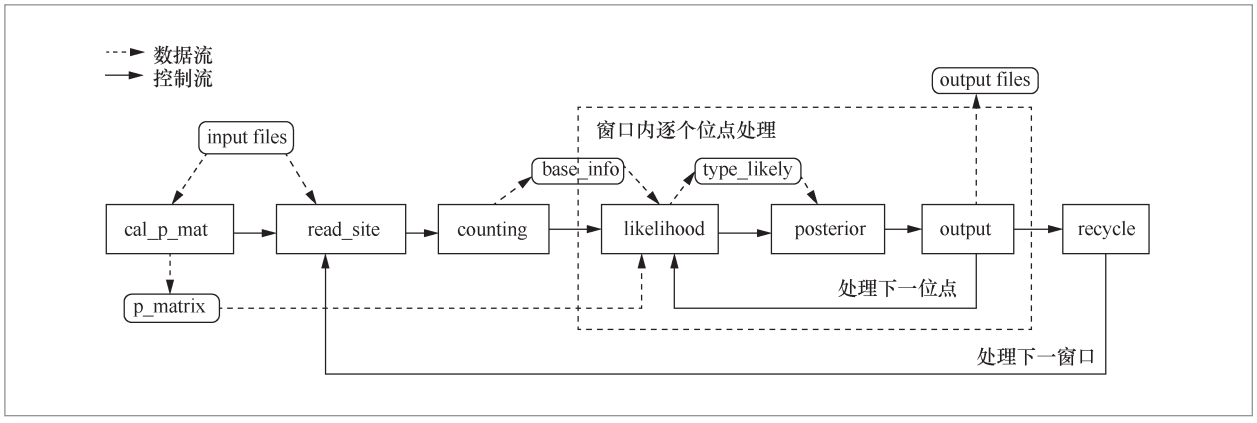


图3 SOAPsnp 程序流程

的碱基类型)的碱基的数量，并生成校准矩阵p_matrix，用于调整概率计算中的测序质量分数。其他6个模块通过多轮处理，完成对全基因组的SNP检测，其中每轮会处理一个窗口：首先，read_site模块从输入文件中加载固定数量的位点（一个窗口宽度）；然后，counting模块收集碱基信息，并保存到base_info内，在每个窗口中，逐个位点依次进行SNP检测；likelihood模块以base_info和p_matrix为输入，计算每个位点每种基因型为SNP的概率，并输出概率值type_likely；posterior模块根据贝叶斯模型计算每种基因型的后验概率，这里将基因型概率以及读段和参考基因组之间的SNP估计值作为先验概率，后验概率计算完成后，当前位点的检测结果被输出到文件，然后从likelihood模块开始，处理当前窗口的下一位点；recycle模块负责窗口切换，即处理跨窗口的读段，并为下一窗口初始化缓冲区。

p_matrix是一个浮点类型的四维矩阵，规模为256×256×4×4，大小为8 MB。矩阵的4个维度分别对应碱基质量分数、读段中的偏移位置、参考基因组碱基类型以及读段中的碱基类型。p_matrix矩阵被用于在似然计算中校正测序的质量分数。base_info是一个规模为4×64×256×2的矩阵，存储了碱基类型、测序质量分数、读段中的偏移坐标和读段的方向等信息。base_info是一个计数矩阵，矩阵中的一个位置对应一种四元组合，矩阵中记录的数值就是满足相应四元组合的碱基的数量。

4 并行优化方法

4.1 序列比对算法的并行优化

利用BWA进行序列比对主要包括生成参考基因组索引和序列比对两个步骤。其中，对于给定的基因组，参考基因组的索引仅需在比对前生成一次，可以被重复使用，因此，研究生成索引过程的并行化对于序列比对来说意义不大，其起到的加速效果十分有限。本文主要研究的内容是将序列比对步骤并行化。

序列比对步骤具体包括读入参考基因组索引、读入短读序列、序列比对和输出比对结果4个子步骤，下面分别介绍各个步骤的并行化方法。

（1）读入参考基因组索引

在进行比对之前，首先必须获得短读序列和参考基因组。BWA虽然实现了多线程比对，但是读入参考序列依然是单线程实现，而且BWA读入索引的方法相对简单。在进行多线程比对时，每个线程会从主线程获得一个指向索引数据结构的指针。这种方法虽然简单，但是行之有效，因为仅需几秒钟就可以将人类基因组索引读入内存中。

多线程可以共享内存空间，但是对于跨节点的MPI程序来说，各个进程的内存空间相互独立，如果仍使用BWA原本的方法，那么需要每个进程分别读一次索引文件，效率低下，而且，当进程规模较大时，会对文件系统造成巨大的压力，严重影响程序性能。

基于以上考虑，本文使用主从模式读取索引。首先指定一个进程为主进程，由主进程将参考基因组索引读入内存，然后再用MPI函数将索引广播到所有其他的进程，这样每个进程都有了索引的一个副本。

这种方法的好处是只需要一个进程访问索引文件即可，缺点是在主进程读入索引时，其他进程都处于空闲状态。不过主进程读取索引文件的速度很快，而且广播可以在$\lg n$（n为进程数量）时间内完成，因此这是一种有效的方法。当大规模系统的互联带宽较高时，这种方法的优势更加明显。

（2）读入短读序列

读入索引后，需要读入进行比对的短

读序列。BWA支持FASTQ格式的读段文件。受限于内存空间，当文件中读段数量较大时，BWA会分批进行迭代处理，直到所有序列比对完毕。类似于读入参考基因组索引，BWA在读入短读序列时也采用单线程方式实现。读入序列时，BWA会一直读取，直到识别到FASTQ格式的行开头信息（>、@或+字符）。

本文读入短读序列采用类似主从模式的方式，首先由一个指定的进程扫描短读序列的FASTQ文件，获取文件中的序列数量、文件行数等信息，然后根据进程数量对读段进行划分。如p号进程处理序列文件的前l条序列，其中$l=\frac{N}{n}\times(p+1)$，N为文件行数，n为进程数量，p从0开始计数。FASTQ文件格式规定，每4行代表一条序列，因此，可以很容易地从序列标号计算出相应的文件行号。主进程每次跳过l条序列时，会将读取文件的位置发给相应的进程，这样每个进程都获得了自己读取文件的位置，也就完成了序列分发。这个过程实际上相当于为短读序列文件建立了一个索引，只是这个索引并未被存储下来。

（3）序列比对

经过前两个步骤，各进程已经有了一份参考基因组索引副本以及分配给本进程的读段，可以进行序列比对。由于序列之间不存在依赖性，各进程间的序列比对是相互独立的，在每个进程内部，BWA本身实现了多线程比对，进一步提高了比对的并发度和速度。

（4）输出比对结果

并行程序的多进程间无法共享文件指针、变量和内存，因此只能每个进程输出单独的文件。各短读序列之间没有相关性，当所有序列的比对都完成后，可以通过简单的shell命令将所有结果文件直接合并成一个总的文件。

另一种方案是采用主从模式，由一个进程负责写文件，其他进程将比对结果发给写文件进程，但是这样会对写文件进程造成很大的压力，因为在各进程并行比对且进程规模较大时，所有进程都使用一个I/O通道，非常拥挤，这会成为系统的瓶颈，影响整个软件的性能。因此，本文采用每个进程单独输出文件的方式。

4.2 SNP检测算法优化

（1）程序热点分析

首先，为了找到SOAPsnp程序的性能瓶颈，笔者分别测试了7个模块的运行时间，发现likelihood和recycle两个模块耗时较长，分别占总时间的59%和30%，排在第三位的是output模块，占比为8.3%。

然后笔者利用性能分析工具进行测试，测试结果显示，likelihood模块花费了大量时间在访问主存，特别是对base_info的访问。base_info用于存储比对的碱基，likelihood模块需要遍历base_info来检测SNP。recycle模块负责初始化下一窗口，处理跨窗口的读段，也包括复制base_info信息。

（2）四维矩阵压缩降维优化

在进行SNP检测时，各位点是相互独立的，在每个位点的计算中，base_info矩阵的每个元素都会被访问一次，将产生131 072次访存（4×64×256×2）。也就是说，整个人类基因组的32亿个位点将产生4.19×10^{14}次访存。

通过分析base_info，笔者发现这是一个高度稀疏的矩阵。base_info的每个元素代表对四元组合(碱基，测序质量分数，读段中的偏移，读段方向)的计数，初始值为0。在碱基中，每发现一个满足特定四元组合的碱基，就会将相应元素增加1。考虑到每个位点比对的碱基数量主要受测序深

度的影响，而测序深度通常低于100层，base_info有131 072个元素，那么base_info矩阵的最大非零比例为100/131 072，也就是0.076%。从这些分析可以看出，base_info矩阵99.9%以上的元素为0，对于计算而言毫无意义，只有不到0.1%的访存是有效的，但是算法仍会遍历整个矩阵，这造成了大量的无用计算。

基于base_info的高度稀疏性，为提高访存效率，笔者直接使用一维数组base_array替代四维矩阵base_info。由于碱基类型、读段方向、测序质量分数和读段中的偏移这些信息的最大值都是确定的，直接通过位操作将这些信息封装在一个32位的字中。在counting模块中，直接将比对的碱基信息依次存储到base_array中，从而避免对base_info矩阵的多次访问，提高程序的空间局部性和cache命中率。

采用一维数组替换四维矩阵后，likelihood计算模块由原来的5层循环变为1层循环（见**表1**），大大提高了计算效率。另外，放弃使用base_info矩阵后，recycle模块复制的数据也减少了很多，不到原来的0.1%。

（3）基于快表的去冗计算

likelihood模块最耗时的部分是更新type_likely的数据结构。在算法中，每次更新type_likely需要分别计算10种基因型（见**表2**）的概率值。每个碱基位点需要计算一次type_likely，对于人类基因组而言，需要访问type_likely数十亿次。每次更新type_likely时，需要访问p_matrix矩阵两次，取出相应数值，再进行一个对数操作来计算概率，这些操作的时钟周期都较长，且type_likely更新的次数很高。

为提高type_likely的计算速度，笔者采用了一种用空间换时间的策略。首先，笔者观察到type_likely每次更新只涉及10种基因型，数量固定，而访问的p_matrix矩阵值和规模也是固定的。因此，对于p_matrix矩阵的每个元素，可以提前计算出10种基因型对应的10个值，并将其保存到内存中，当type_likely需要访问p_matrix的某个元素时，直接从预先算好的矩阵（预计算矩阵）中读取。p_matrx矩阵的大小为8 MB，对应10种基因型，则预计算矩阵的大小为80 MB。虽然这样牺牲了一定的内存空间，但是可以将原来的两次访存及一次对数操作变为一次访存操作，降低了算法的复杂度。

（4）I/O压缩策略

对SOAPsnp进行算法改进后，计算速度显著提高。但是随之而来的一个问题是，优化前占程序运行总时间8.3%的输出模块占优化后程序总时间的50%以上，成为新的瓶颈。为提高程序的I/O性能，笔者进一步分析了SOAPsnp程序。输入文件主要为比对完成的读段，是其他软件的输出，因此，暂时不考虑优化。SOAPsnp的输出为一个17列的文本文件，如**图4**所示。

通过观察SOAPsnp的输出文件发现，每一列都存在很多重复的值，可以针对每列采取不同的压缩方式。其中第1列为参考序列名称，**图4**中的chr22是染色体名称，这一列的值完全相同，因此只需要存储一次。第2列为碱基在参考序列上的偏移，每行递增，因此也只需要存储第一个值。对于存储碱基值的第3、4、6列，由于SNP的

表1 优化前后的碱基信息遍历算法

算法1 遍历base_info的原始算法

```
for o_base = 0 to 3
  for q_score = q_max to 0
    for coord = 0 to read_length
      for strand = 0 to 1
        for k = 0 to base_info[index]
          // likelihood computation
          ……
        end for
      end for
    end for
  end for
end for
```

算法2 遍历base_array的优化算法

```
for ix=0 to base_array_size
  //likelihood computation
  ……
end for
```

占比非常低，因此这3列的绝大部分值是相同的，可以合并存储，只需要额外存储那些是SNP的位点。与基因型相关的列都是高度稀疏的，只需要存储非零值即可。第11、12、13、15、17列为默认值，只有存在SNP时该值才发生变化。因此，这几列也只需要存储那些不同的值和一个默认值即可。其他列则根据特性，可分别采用游程编码或字典等方式进行压缩。

4.3 SNP检测算法的并行优化

各个位点的SNP检测是相互独立的，具有天然的并行性，笔者采用OpenMP和MPI技术实现了多级并行SNP检测。在窗口内部，各位点之间采用基于OpenMP的多线程并行，每个线程计算一个位点，这样可以利用多线程共享数据的特性，降低内存压力。此外，为提高SNP检测速度，笔者还利用MPI技术实现了多窗口间的并行，不同的进程运行在不同的计算节点上，分别处理不同的窗口。对于跨窗口的读段，在开始每个窗口的计算前，相邻窗口会进行两轮通信，第一次通信是确认每个窗口的位置，确定没有重叠和遗漏；第二次通信是交换跨窗口的读段的信息，相当于**图3**中的recycle模块，对窗口进行初始化，这样就可以保证读段信息的完整性和SNP检测的正确性。

表 2 基因型的表示方法

基因型	计算机表示	索引
AA	0000	0
AC	0001	1
AG	0010	2
AT	0011	3
CC	0101	5
CG	0110	6
CT	0111	7
GG	1010	10
GT	1011	11
TT	1111	15

4.4 变异检测分析流水线I/O整合优化

在基因组变异检测分析流水线中，序列比对和SNP检测是相邻的两个步骤，序列比对软件首先将比对结果输出到文件，由于SNP检测需要基于排序好的读段进行，因此，还需要对输出的比对结果进行排序，并行输出排序好的比对文件，SNP检测以排好序的比对文件为输入进行分析。以上流程需要3次大规模读操作和3次大规模写操作，而基因组规模一般较大，且高通量测序通常需要测多层，以保证数据的鲁棒性，因此，比对、排序和SNP检测对I/O造成了巨大压力，特别是当程序不断优化，计算时间不断缩短，而数据规模不断增长

列号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	chr22	1	G	G	1	G	0	0	0	T	0	0	0	0	1.00000	255.000	0
	chr22	2	A	A	1	A	0	0	0	T	0	0	0	0	1.00000	255.000	0
	chr22	3	T	T	1	T	0	0	0	T	0	0	0	0	1.00000	255.000	0
	chr22	4	C	C	1	C	0	0	0	T	0	0	0	0	1.00000	255.000	0
	chr22	5	T	T	1	T	0	0	0	T	0	0	0	0	1.00000	255.000	0
	chr22	6	G	G	1	G	0	0	0	T	0	0	0	0	1.00000	255.000	0
	chr22	7	A	A	1	A	0	0	0	T	0	0	0	0	1.00000	255.000	0
	chr22	8	T	T	1	T	0	0	0	T	0	0	0	0	1.00000	255.000	0
	chr22	9	A	A	1	A	0	0	0	T	0	0	0	0	1.00000	255.000	0

图 4 SOAPsnp 输出文件格式

时，I/O成为程序的主要瓶颈。

针对这一问题，本文提出了一种I/O整合策略。当序列比对完成后，将比对结果暂时保存在内存中，对结果进行排序，然后直接基于排序后的结果进行SNP检测，这样可以避免两轮I/O操作。另外，程序也会正常输出比对结果，用于后续其他分析。这样一来，只需要读入一次参考基因组和序列文件，即可完成比对和SNP检测两个步骤，大大降低了I/O压力，提高了数据处理的速度。

5 测试分析

5.1 测试环境与数据

本文的测试主要在“天河二号”超级计算机上进行，节点配置见**表3**。

实验采用的物种为人类，参考基因组版本为hg38，读段数据包括真实数据和模拟数据，真实数据由Illumina测序仪产生，模拟数据由wgsim生成，长度均为100碱基对（base pair，BP）。

5.2 序列比对测试分析

（1）强可扩展性

本节通过对500万条序列进行比对来测试程序的强可扩展性，进程数分别为1个、2个、4个、8个、16个、32个、64个。测试数据见**表4**。

表 3 “天河二号”超级计算机节点配置

名称	详情
CPU	Intel Xeon E5-2692 v2
接口数/核数/线程数	2/12/2
时钟频率/GHz	2.20
L1/L2/L3, Cache/KB	32/256/30,720
内存大小/GB	64

如**图5**所示，若保持问题规模不变，加入更多的处理器，程序处理时间接近于线性减少，这证明程序具有良好的可扩展性。虽然每个进程都要读入一份参考索引，但是采取进程广播的形式来分发参考索引并不会让程序总体处理时间有明显增加，而总体的程序处理时间大大减少，运行时间接近于理想情况的1/*n*，表现出良好的强可扩展性。

（2）弱可扩展性

在弱可扩展性测试中，令每个进程处理50万条序列，进程数分别为1个、2个、4个、8个、16个、32个、64个，数据规模随进程数的增加同比扩大。测试数据见**表5**。

如**图6**所示，若在进程数增加的同时同比扩大数据规模，每个进程平均处理50万条序列，程序运行时间保持稳定。从单进程到多进程，处理数据的时间会有明显增加，这主要是因为多个进程在输入、输出数据时需要更多时间，并且进程之间需要通信，这导致了程序执行时间的增加。在程序并行执行时，平均每个进程运行50万条序列，运行时间波动不大，基本保持稳定。这说明，在增加数据规模的同时增加进程数可以使处理机在相同的时间内处理更多的数据，程序的弱可扩展性良好。

5.3 SNP检测测试分析

（1）算法优化效果测试

本文首先测试了各种算法优化策略的加速效果，以SOAPsnp单线程程序为基准，逐一添加优化策略，并测试运行时间，如**图7**所示。在**图7**中，横轴的SOAPsnp表示原始算法，o1表示base_info压缩优化，o1+o2表示在o1的基础上添加快表优化，o1+o2+o3表示在o1+o2的基础上进行了输出压缩优化。

从**图7**可以看到，效果最显著的是base_info压缩优化，因为压缩后不仅大大减少了计算量，还降低了空间开销，一次优

化降低了likelihood和recycle两个模块的运行时间。快表优化是对likelihood计算核心的优化，以时间换空间，起到了很好的效果。输出压缩优化则在计算模块加速充分的情况下，进一步降低了程序运行时间，提高了处理速度。

（2）并行程序可扩展性

本文利用OpenMP+MPI的并行技术实现了多级并行。图8展示的是程序的强可扩展性。以128个节点的性能为基准，程序运行时间近似呈线性下降，表明程序具有良好的可扩展性。图9展示的是程序弱可扩展性的并行效率，测试数据集的大小随节点规模同比扩大，最大数据规模达到了542 GB，从图9可以看出，程序并行效率随着计算规模的扩大呈下降趋势，虽然笔者对程序的通信和输出进行了优化，但是当数据量和进程规模较大时，扫描读段文件、进程通信和结果输出所占的比例会不断提高，计算占比会相对降低，导致程序的并行效率降低，但是总体并行效率仍保持在60%以上。

6 结束语

本文选取基因组大数据变异检测中最耗时的两个环节——序列比对和SNP检测，首先从算法角度进行了优化和改进，然后利用OpenMP、MPI等并行技术实现了多级并行处理。测试表明，并行优化后的算法具有良好的并行性能和可扩展性，为快速处理高通量测序技术带来的海量基因组大数据提供了有力的支持。

参考文献:

[1] STEPHENS Z D, LEE S Y, FAGHRI F, et al. Big data: astronomical or genomical?[J].

表 4 同一读段长度下不同进程数的程序运行时间

进程数/个	1	2	4	8	16	32	64
时间/s	954	485	250	132	72	50	34

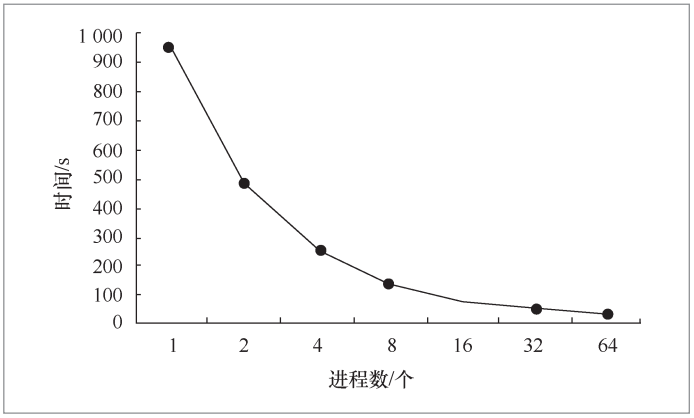


图 5 序列比对程序强可扩展性

表 5 不同进程数和读段长度下程序运行时间

进程数/个	1	2	4	8	16	32	64
时间/s	104	125	126	122	126	127	128

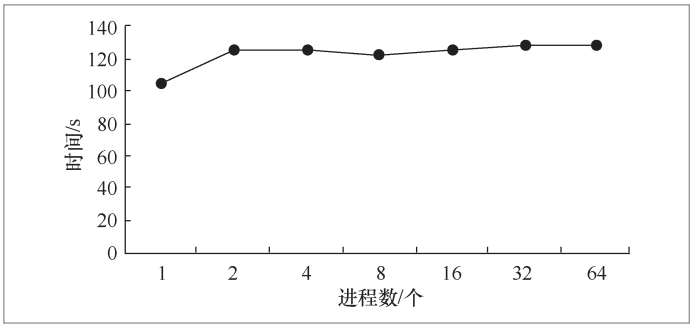


图 6 序列比对程序的弱可扩展性

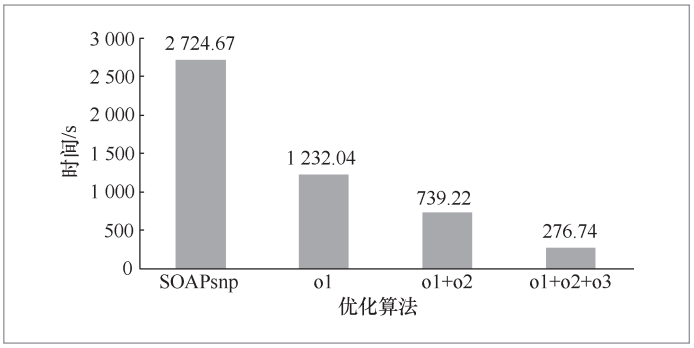


图 7 算法优化加速效果

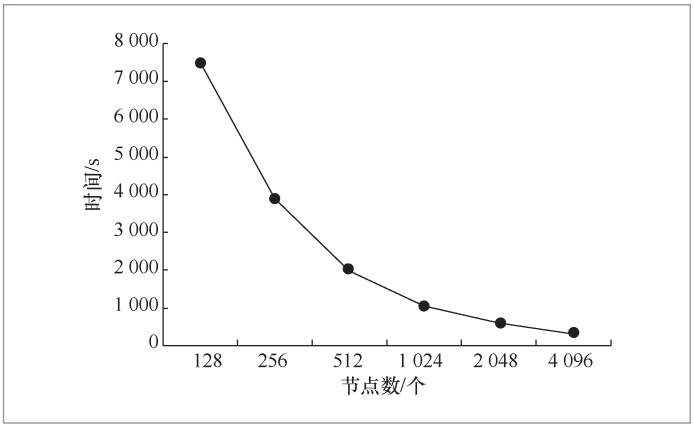


图 8 程序强可扩展性

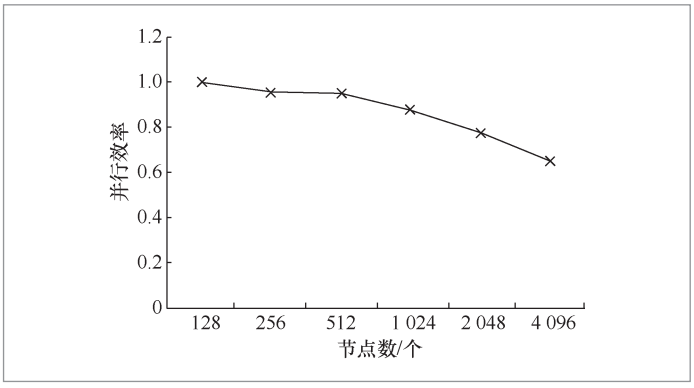


图 9 程序弱可扩展并行效率

PLoS Biology, 2015, 13(7): e1002195.

[2] MARX V. Biology: the big challenges of big data[J]. Nature, 2013, 498(7453): 255-260.

[3] KATHIRESAN N, TEMANNI R, ALMABRAZI H, et al. Accelerating next generation sequencing data analysis with system level optimizations[J]. Scientific Reports, 2017, 7(1): 9058.

[4] LI H, HOMER N. A survey of sequence alignment algorithms for next-generation sequencing[J]. Briefings in Bioinformatics, 2010, 11(5): 473-483.

[5] ALTSCHUL S F, GISH W, MILLER W, et al. Basic local alignment search tool[J]. Journal of Molecular Biology, 1990, 215(3): 403-410.

[6] LI R, LI Y, KRISTIANSEN K, et al. SOAP: short oligonucleotide alignment program[J]. Bioinformatics, 2008, 24(5): 713-714.

[7] JIANG H, WONG W H. SeqMap: mapping massive amount of oligonucleotides to the genome[J]. Bioinformatics, 2008, 24(20): 2395-2396.

[8] SMITH A D, XUAN Z, ZHANG M Q. Using quality scores and longer reads improves accuracy of Solexa read mapping[J]. BMC Bioinform, 2008, 9(1): 128.

[9] HOMER N, MERRIMAN B, NELSON S F. BFAST: an alignment tool for large scale genome resequencing[J]. PLoS One, 2009, 4(11): e7767.

[10] SCHATZ M C. CloudBurst: highly sensitive read mapping with MapReduce[J]. Bioinformatics, 2009, 25(11): 1363-1369.

[11] CHEN Y, SOUAIAIA T, CHEN T. PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds[J]. Bioinformatics, 2009, 25(19): 2514-2521.

[12] CLEMENT N L, SNELL Q, CLEMENT M J, et al. The GNUMAP algorithm: unbiased probabilistic mapping of oligonucleotides from next generation sequencing[J]. Bioinformatics, 2010, 26(1): 38-45.

[13] FERRAGINA P, MANZINI G. Opportunistic data structures with applications[C]//The 41st Annual Symposium on Foundations of Computer Science. Piscataway: IEEE Press, 2000: 390-398.

[14] LI H, DURBIN R. Fast and accurate short read alignment with Burrows-Wheeler transform[J]. Bioinformatics, 2009, 25(14): 1754-1760.

[15] LANGMEAD B, TRAPNELL C, POP M, et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome[J]. Genome Biology, 2009, 10(3).

[16] LI R, YU C, LI Y, et al. SOAP2: an improved ultrafast tool for short read alignment[J]. Bioinformatics, 2009, 25(15): 1966-1967.

[17] LI R, LI Y, FANG X. SNP detection for massively parallel whole-genome resequencing[J]. Genome Research, 2009, 19(6): 1124-1132.

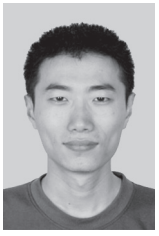
[18] TANG J, LEUNISSEN J A M, VOORRIPS R E. HaploSNPer: a web-based allele and SNP

detection tool[J]. BMC Genetics, 2008, 9(1).
[19] DEREEPER A, NICOLAS S, LE C L. SNiPlay: a web-based tool for detection, management and analysis of SNPs. Application to grapevine diversity projects[J]. BMC Bioinformatics, 2011, 12(1).
[20] TANG J, VOSMAN B, VOORRIPS R E. QualitySNP: a pipeline for detecting single nucleotide polymorphisms and insertions/deletions in EST data from diploid and polyploid species[J]. BMC Bioinformatics, 2006, 7(1).
[21] LI H, HANDSAKER B, WYSOKER A. The sequence alignment/map format and SAMtools[J]. Bioinformatics, 2009, 25(16): 2078-2079.
[22] DEPRISTO M A, BANKS E, POPLIN R. A framework for variation discovery and genotyping using next-generation DNA sequencing data[J]. Nature Genetics, 2011, 43(5): 491-498.
[23] RACZY C, PETROVSKI R, SAUNDERS C T. Isaac: ultra-fast whole-genome secondary analysis on Illumina sequencing platforms[J]. Bioinformatics, 2013, 29(16): 2041-2043.
[24] WEI Z, WANG W, HU P, et al. SNVer: a statistical tool for variant calling in analysis of pooled or individual next-generation sequencing data[J]. Nucleic Acids Research, 2011, 39(19): 132.
[25] KOBOLDT D C, CHEN K, WYLIE T, et al. VarScan: variant detection in massively parallel sequencing of individual and pooled samples[J]. Bioinformatics, 2009, 25(17): 2283-2285.
[26] KOBOLDT D C, ZHANG Q, LARSON D E, et al. VarScan2: somatic mutation and copy number alteration discovery in cancer by exome sequencing[J]. Genome Research, 2012, 22(3): 568-576.
[27] LANGMEAD B, SCHATZ M C, LIN J. Searching for SNPs with cloud computing[J]. Genome Biology, 2009, 10(11).
[28] ZHAO S, PRENGER K, SMITH L. Rainbow: a tool for large-scale whole-genome sequencing data analysis using cloud computing[J]. BMC Genomics, 2013, 14(1).
[29] LU M, ZHAO J, LUO Q. GSNP: a DNA single-nucleotide polymorphism detection system with GPU acceleration[C]// International Conference on Parallel Processing. [S.l.:s.n.], 2011: 592-601.
[30] 彭绍亮, 牛琦, 李肯立, 等. CPU-MIC异构并行架构下基于大规模频繁子图挖掘的药物发现算法[J]. 大数据, 2019, 5(2): 89-103.
PENG S L, NIU Q, LI K L, et al. A scalable CPU-MIC coordinated drug-finding tool by frequency subgraph mining[J]. Big Data Research, 2019, 5(2): 89-103.
[31] 彭绍亮, 杨顺云, 孙哲, 等. 生物效应大数据评估聚类算法的并行优化[J]. 大数据, 2018, 4(3): 24-36.
PENG S L, YANG S Y, SUN Z, et al. Parallel optimization for clustering algorithm of large-scale biological effect evaluation[J]. Big Data Research, 2018, 4(3): 24-36.
[32] SCHINDLER M. A fast block-sorting algorithm for lossless data compression[C]// The Conference on Data Compression. Piscataway: IEEE Press, 1997.

作者简介

崔英博(1989-),男,博士,国防科技大学计算机学院助理研究员,主要研究方向为高性能计算、生物大数据挖掘等。

黄春(1973-),女,博士,国防科技大学计算机学院研究员,主要研究方向为高性能计算系统、并行编译、并行编程和高性能数学库等。



唐滔(1984-),男,博士,国防科技大学计算机学院副研究员,主要研究方向为编译器、并行计算和高性能计算。

杨灿群(1968-),男,博士,国防科技大学计算机学院研究员,主要研究方向为高性能计算、并行编程等。

廖湘科(1963-),男,博士,国防科技大学计算机学院研究员,主要研究方向为高性能计算、系统软件等。

彭绍亮(1979-),男,博士,湖南大学信息科学与工程学院教授,国家超级计算长沙中心副主任,主要研究方向为高性能计算、生物信息、大数据挖掘、区块链等。

收稿日期: 2020-06-30

基金项目: 国家重点研发计划基金资助项目(No. 2018YFB0204301, No. 2017YFB0202602);国家自然科学基金资助项目(No. 61772543, No. 61972408)

Foundation Items: The National Key Research and Development Program of China (No. 2018YFB0204301, No. 2017YFB0202602), The National Natural Science Foundation of China (No. 61772543, No. 61972408)

医疗大数据在学习型健康医疗系统中的应用

柴扬帆[1,2]，孔桂兰[1]，张路霞[1]
1. 北京大学健康医疗大数据国家研究院，北京 100191；
2. 北京大学公共卫生学院，北京 100191

摘要
将医疗大数据应用于旨在加快知识生成和临床转化应用的学习型健康医疗系统（LHS）中，满足患者和医疗决策者的知识需求，有助于推动精准医学的发展。在系统阐述医疗大数据与LHS发展现状的基础上，结合LHS的典型应用案例，重点分析医疗大数据在LHS中的应用特点及面临的挑战。最后总结了我国发展LHS面临的挑战，并对未来进行了展望。

关键词
医疗大数据；学习型健康医疗系统；医疗决策

中图分类号：R-1 文献标识码：A doi: 10.11959/j.issn.2096-0271.2020042

Application of medical big data in learning health system

CHAI Yangfan[1,2], KONG Guilan[1], ZHANG Luxia[1]
1. National Institute of Health Data Science at Peking University, Beijing 100191, China
2. School of Public Health, Peking University, Beijing 100191, China

Abstract
The learning health system (LHS) aims at accelerating the process of knowledge generation, transformation and application in clinical practice. Applying medical big data in LHS to meet the knowledge needs of patients and healthcare decision makers would help to promote the development of precision medicine. Firstly, the current status of medical big data and LHS were reviewed, then the characteristics and challenges of applying medical big data in LHS were analyzed by refering to some typical application cases. Finally, the challenges faced by LHS in China were addressed and the prospect of applying medical big data to LHS in the future was provided.

Key words
medical big data, learning health system, medical decision-making