Gene Panel 流程的并行设计与优化研究

王元戎[1),2)] 曾 平[1),2)] 臧大伟[1)] 谭光明[1)] 孙凝晖[1)]

[1)](中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190)

[2)](中国科学院大学 北京 100049)

摘 要 随着二代测序技术的快速发展,基因测序成本迅速下降,这导致基因数据的爆炸式增长,基因数据分析工具逐渐无法满足如此大规模的数据分析需求.一方面,基因数据分析工具大多仍为串行执行,无法有效地利用多核结构提升性能并导致计算资源的严重浪费;另一方面,由于前期设计和开发的局限性,分析工具所依赖的底层算法库不能兼顾高性能与友好的用户接口.Gene Panel 是当前主流的面向癌症检测的基因数据分析流程,它也是由多种基因数据分析工具组成的.该文面向 Gene Panel 流程:(1)设计并实现了一套全新的并行 Gene Panel 基因数据分析流程,通过数据并行和任务并行两种主要并行手段并结合负载均衡等其他优化方法,有效地提升了多核平台的资源利用率,并获得了 4～7 倍的整体加速比;(2)设计并实现了一种接口友好的高性能基因数据分析底层库 HCC.由于相似的算法特征,该文的优化方法同样适用于除 Gene Panel 外的其他测序流程.

关键词 大数据;Gene Panel;并行优化;负载均衡;底层库优化

中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2019.02429

Design and Optimization of Parallel Gene Panel Process

WANG Yuan-Rong[1),2)] ZENG Ping[1),2)] ZANG Da-Wei[1)] TAN Guang-Ming[1)] SUN Ning-Hui[1)]

[1)](*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

[2)](*University of Chinese Academy of Sciences, Beijing* 100049)

Abstract Nowadays, the costs of genome sequencing are rapidly reduced, with the rapid development of the next generation sequencing (NGS) technology. This is bringing a bright future to clinical diagnosis and precision medicine. However, the improvement of genome sequencing technology is resulting in gene data explosion at the same time. In fact, sequencing is just the first step in the clinical disease detection process. The sequenced gene data outputted from the sequencer need to be further processed by several analysis tools. The genetic data analysis tools are gradually unable to meet such a large-scale data analysis demand. On one hand, most of the genetic data analysis tools are still executing serially, thus they can not effectively make use of the multi-core architecture to help improve performance, and this also leads to a serious waste of computing resources at the same time; On the other hand, due to the limitations of previous design and development, the interface of the underlying genetic data analysis library used by some tools is unable to take both high performance and user-friendly interfaces into account. This paper studied Gene Panel, the most commonly used genetic data analysis process specially designed for cancer detection. We profiled Gene Panel process running on a multi-core Intel Xeon server and observed that more than 90% execution time is cost on serial execution of the analysis tools, and

收稿日期:2017-12-01;在线收稿日期:2018-08-31.本课题得到科技部国家重点研发计划(2016YFB0200300,2016YFB0201305,2016YFB0200504,2016YFB0200803,2016YFB0200204)和中国科学院战略性先导科技专项(XDB24050300)资助.**王元戎**,博士研究生,主要研究方向为生物大数据加速器设计.E-mail:wangyuanrong@ncic.ac.cn.**曾 平**,硕士,主要研究方向为生物大数据并行计算框架.**臧大伟**,博士,助理研究员,中国计算机学会(CCF)会员,主要研究方向为计算机系统结构.**谭光明**,博士,研究员,博士生导师,中国计算机学会(CCF)会员,主要研究领域为并行算法与体系结构.**孙凝晖**,博士,研究员,博士生导师,中国计算机学会(CCF)会员,主要研究领域为并行处理器体系结构、分布式操作系统、高性能计算等.

thus in most of the execution time, only one core of the multi-core server is working. Based on the Gene Panel process, we designed and implemented a novel Gene Panel data analysis process with massive data-level and task-level parallelism. The mapping tool is used in the first step of the data analysis process and it has been written in multithreading model. We performed SIMD optimizations on it as operations on its elements are easily vectorized. For the tools used in subsequent steps, we adopted the multitasking way that each software thread running on each core processes part of the whole task in parallel. To improve the performance and usability of the current underlying library for genetic analysis, we designed and implemented an interface-friendly library for high-performance genetic data analysis. Besides, we combined our new Gene Panel process with other optimization methods, such as load balancing and inter-algorithm overlap. They also effectively raise the resource utilization rate of the multi-core platform. The new parallel Gene Panel process proposed in this paper achieves a 4—7 times overall speedup over the original process. In general, our contributions are as follows: (1) We made a deep investigation on the algorithms of Gene Panel and found out their potential parallel characteristics; (2) We designed and implemented a novel Gene Panel process with both parallelizing techniques and other optimization methods; (3) We modified and implemented the underlying genetic analysis library with both user-friendly interface and high performance; (4) As most analysis tools used in Gene Panel process are also components or share similar algorithmic characteristics as the components of other genetic analysis processes, such as Whole Gene Sequencing (WGS) and Whole Exome Sequencing (WES), the optimization methods proposed in this paper are also applicable to other genetic data analysis processes.

Keywords big data; Gene Panel; parallel optimization; load balancing; underlying library optimization

1 引 言

随着人类基因组计划的完成，我们已经获得了一套比较完整的人类基因参考序列. 对于个体，我们可以以该参考序列为基准，通过成本低廉的测序技术，并结合一些基因比对和拼接工具，来得到个体的基因序列. 基因是遗传的密码，通过比较个体基因序列与参考序列的差异，可以检测患病风险并有目的地指导医学治疗. Gene Panel 是一套最常用的面向癌症检测的基因数据分析流程，是基于 GATK① 标准基因分析流程定制修改而来的. Panel[1] 是生物学家根据实验分析得到的一些染色体位置集合，这些位置的基因发生变异会大概率影响某些疾病的发生. 因此，可以通过分析这些区间的基因突变情况来检测某种疾病是否存在发病的风险. Gene Panel 主要对 panel 文件定义的区间进行分析，发现变异位点. 它通常用在疾病检测如癌症检测中. 人类参考序列有大约 30 亿②位点；另一方面，为保证准确率，Gene Panel 流程的数据覆盖深度一般为 2000x 以上③，这些都大大增加了待处理的数据量.

而随着高通量测序的发展，测序速度从 1980 年的每天一万碱基提升到现在的每天几万亿碱基，测序成本也从 2001 年的 1 亿美元降到现在不足 1 千美元④，这是当初基因组计划启动时无法想象的. 低廉的测序成本使得个体基因测序以及疾病预测成为可能. 随着基因测序技术的不断发展，来自全球的基因分析数据中心的基因数据呈现爆炸式增长. 而对基因数据的处理能力却远远落后于数据产生的速度. 为了应对日益增长的基因大数据处理需求并进一步降低测序成本，基因数据分析流程的性能亟待提升.

本文以 Gene Panel 基因数据分析流程为研究对象，发现现有流程中的算法在多核处理器上的低效性严重影响了流程的性能. 其主要体现在如下方面：

(1) 流程 92%的执行时间都花费在串行执行的算法上面，造成很大的多核资源浪费.

① https://software.broadinstitute.org/gatk/
② https://www.genome.gov/11006943
③ https://blog.genohub.com/2016/10/24/
④ https://www.genome.gov/27541954/dna-sequencing-costs-data/

(2) 流程所依赖的基因分析底层库无法兼顾高性能与友好的用户接口：HTSJDK① 提供了友好的用户接口却牺牲了性能；HTSLIB② 虽然性能较高，但接口易用性差，学习成本高.

(3) 由于数据分布不均的特点，对流程算法进行简单的并行化会带来负载不均衡等问题，进一步影响性能.

为此，本文设计、实现并评估了一套面向 Gene Panel 流程的并行加速方法. 通过数据并行和任务并行两种主要手段，同时结合负载均衡等其他优化方法，有效地提升了多核平台的资源利用率，与原有 Gene Panel 流程相比，整体性能提升了 4～7 倍.

本文的贡献如下：

(1) 深入分析了 Gene Panel 流程中算法潜在的并行特征，设计并实现了一套基于数据和任务并行的全新的并行 Gene Panel 基因数据分析流程.

(2) 设计并实现了一种接口友好的高性能基因数据分析底层算法库 HCC.

(3) 提出并实现了面向 Gene Panel 流程的负载均衡及利用内存文件系统加速中间结果 IO 等优化手段.

本文第 2 节介绍国内外相关工作研究；第 3 节剖析 Gene Panel 流程的性能瓶颈；第 4 节描述流程优化过程中采用的并行优化手段；第 5 节描述优化过程中的其他问题以及解决方法；第 6 节给出实验结果，并对实验结果加以分析；最后在第 7 节，对本文加以总结，并指出进一步的工作方向.

2　相关工作

国内外研究机构从算法并行化、体系结构优化、流程并行化以及分布式计算等方面对基因测序流程的优化开展了深入研究.

2.1　算法并行化

最新版的 BWA-MEM[2]、Bowtie 2[3] 等比对算法都支持多线程；进程级并行常用在集群环境中，这方面比较有代表性的是 mpiBLAST[4]. GATK 开发团队和 Intel 团队合作，使用 AVX 指令实现了 Pair-HMM，在原有计算资源相同的基础上对程序进行加速. 对于 BWA-MEM 中的 Smith-Waterman 算法[5]，Li 等人也利用 SSE 对其进行加速[6].

2.2　体系结构优化

异构计算技术从 20 世纪 80 年代中期产生，由于它能经济有效地获取高性能计算能力、可扩展性好、计算资源利用率高、发展潜力巨大，目前已成为并行和分布计算领域中的研究热点之一. 目前流行的解决方案有：可编程门阵列(FPGA)与通用处理器(CPU)混合架构，GPU 与 CPU 混合架构，Intel 集成众核(MIC)与 CPU 混合架构. 早在 2012 年，剑桥大学的 Klus 等人就开发了 BarraCUDA[7]，它是基于 BWA 0.6 版本进行开发的，不仅支持允许空位比对，还可以利用多个 GPU 提升加速效果. 此外 SW-CUDA[8]、CUDASW++[9]，GPU-BLAST[10] 等都采用 GPU 进行加速. Ren 等人也尝试在 FPGA 上针对 Pair-HMM 进行优化，实现了高带宽 IO 和高并行度[11]. 上述体系结构相关的优化工作都是针对流程中的单个算法甚至算法中的某一部分，这是因为某些加速平台的结构对应用是有“偏好”的，比如 GPU 更适合计算密集型应用，而 Gene Panel 流程中多数算法是访存受限的，本文讨论的是对整个 Gene Panel 流程的并行加速，因此 GPU 并不是最合适的平台. FPGA 虽然为定制硬件结构提供了可能，但其对开发人员专业性要求较高，对整个流程进行硬件定制较为困难；另一方面，FPGA 实现侧重在算法专用的并发数据通路定制这一层面，而本文主要着眼于任务划分和并行，二者关注不同层面，是正交的. 同样的，本文探讨单节点内 Gene Panel 的并行优化，如果将基因分析工作移植到高性能集群上，由于负载天然的可并行性(不同任务间无依赖，比如不同节点处理不同人的基因序列)，集群每个节点是互不依赖的，因此，本文的工作同样适用于集群处理系统.

2.3　流程并行化

Churchill DNA 分析流程，提出通过将输入数据切成很多段来实现跨染色体区域的并行[12]. 这种方法能够有效通过在一个定制的高性能集群中将不同分析步骤结合到一个单一的紧密集成的计算管道中，实现资源的有效利用. 在一个 48 核的 Dell R815 服务器上，运行在定制高性能集群上的 Churchill 相对于开启多线程的 GATK 流程有 10 倍加速. 然而，这种解决方案得到的结果难以被第三方检测和评估.

3　流程的性能瓶颈剖析

3.1　Gene Panel 数据分析流程

Gene Panel 数据分析流程如图 1 所示，流程主

① https://samtools.github.io/htsjdk
② http://www.htslib.org

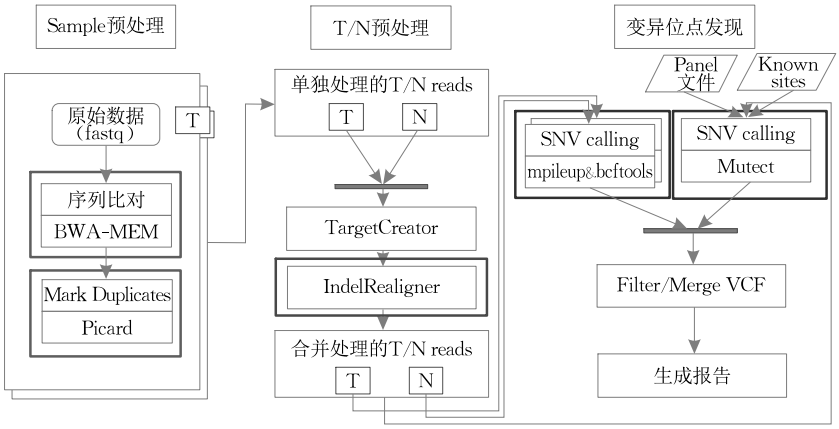


图 1 Gene Panel 流程图

要分为三个部分：Sample 预处理、Tumor/Normal 预处理和变异位点发现(Variant Calling).

3.1.1 Sample 预处理

Sample 预处理阶段是对正常组织基因样本(Normal)和变异组织基因样本(Tumor)分别进行预处理，两者互不依赖.此阶段主要用到 BWA-MEM 和 MarkDuplicate① 两个算法.BWA-MEM 是当前应用最广泛的比对工具，其作用是将测序仪输出的 DNA 片段(read)比对到人类参考遗传序列(Reference)上；BWA-MEM 的输出结果(SAM 文件)经过简单的排序(sort)后，由 MarkDuplicate 对其进行去冗余处理.

MarkDuplicate 的作用是将对结果里冗余的数据进行标记，这样后续步骤中这些被标记的数据不会被处理，从而大大减小流程后续步骤的工作量，进而缩短流程的执行时间.

3.1.2 Tumor/Normal 预处理

Tumor/Normal 预处理需要同时输入 Tumor 和 Normal 样本数据，此步骤的主要作用是对“插入/删除”位点(Indel)所在的局部区域进行重新比对，来降低 Indel 附近的错误率，从而提高变异位点发现的精确度，此步骤采用的是 GATK 工具包中的 IndelRealigner.

3.1.3 变异位点发现

变异位点发现用来判定当前位点是否是一个变异位点(SNP).此部分采用 Samtools② 工具包中的 mpileup 和 mutect 工具分别进行判定，然后将这两个工具发现的变异位点信息合并和过滤，最后生成病人的报告信息.

3.2 流程在多核处理器上执行的低效性

为了优化 Gene Panel 流程的性能，需要先找流程执行的瓶颈所在，我们在双路 24 核心 Intel Xeon 服务器上测量了流程中各部分算法的时间占比，结果如表 1 所示.

表 1 Gene Panel 流程时间分解
(T：Tumor；N：Normal)

流程各阶段算法	耗时/s	占比/%	是否并行
BWA&Sort(T)	525	4.9	是
BWA&Sort(N)	173	1.6	是
Mark Duplicate(T)	1115	10.3	否
Mark Duplicate(N)	363	3.4	否
IndelTargetCreator	175	1.6	是
IndelRealigner	3957	36.7	否
Mutect	2815	26.1	否
Mpileup&bcftool(T)	1098	10.2	否
Mpileup&bcftool(N)	557	5.2	否
Total	10778	100.0	—

从表 1 可以看出，流程中大部分算法都只支持单线程模式，即只能串行执行，并且单线程部分执行时间占总时间的 92%，只有少数算法(如 IndelTargetCreator)支持并行处理模式，从而导致整体流程执行时间较长.另一方面，现有流程多数算法没有充分利用服务器多核以及超线程资源，使得算法与硬件平台结构的不匹配问题严重.

3.3 面向多核处理器的并行算法优化

数据并行. 单指令多数据(SIMD)是一种通过一条指令，同时控制多个处理单元对一组数据(又称“数据向量”)中的每一个元素分别执行相同的操作从而实现空间并行性的技术，例如 Intel 的“SSE”或“AVX”扩展指令集.SIMD 适用于 Smith-Waterman 等计算密集型算法.

任务并行. 任务并行是指通过对计算任务进行粗粒度划分，多个功能部件同时各自处理部分任务，从而协同完成工作，例如多核处理器上的多线程和

① http://broadinstitute.github.io/picard/
② http://samtools.sourceforge.net/

多进程执行模式，但这需要相应的多线程/多进程编程模型来支撑.

事实上，Smith-Waterman 算法数据依赖只存在于相邻元素，对于不相邻元素的处理适合用数据并行；而后续算法基本都以某种基本的数据结构为单位进行遍历处理，这些基本数据结构包括 read 以及位点上 read 堆叠(pileup)等，它们都可以通过对这些基本数据结构，或者包含这些基本数据结构的其他结构(如染色体区间 interval)进行合理划分来实现任务并行. 流程并行设计方法概况如图 2 所示.

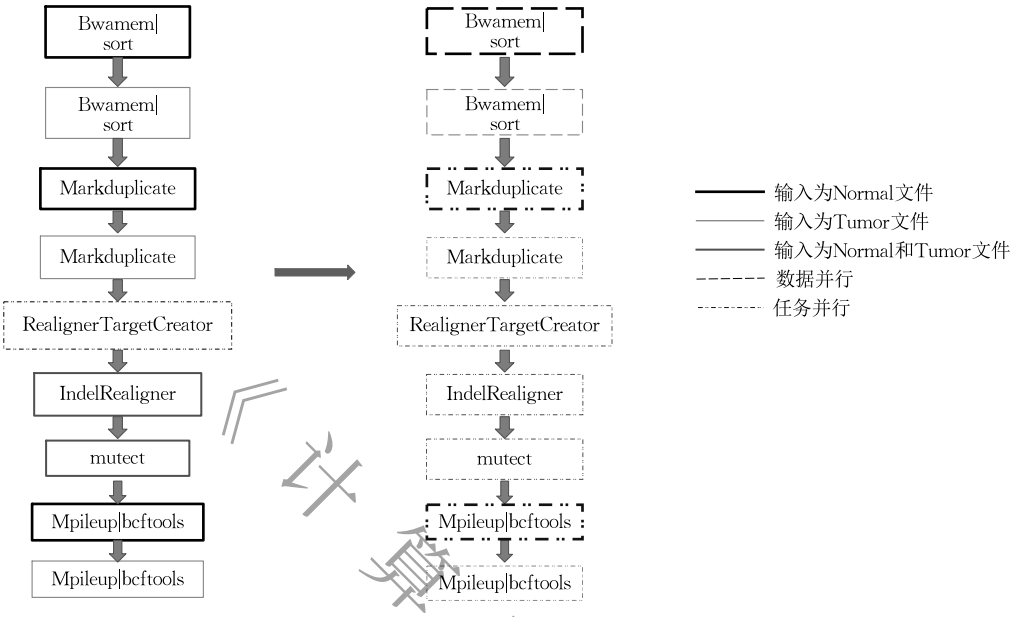


图 2　流程并行设计方法

此外，优化过程还需要解决其他问题，比如：(1)为 SIMD 提供整齐的向量结构；(2)处理任务划分带来的负载不均衡问题；(3)提供接口易用的高性能底层库；以及(4)从整个流程的角度进一步提升流程性能等.

4　流程并行设计

为了挖掘流程中算法的并发度以高效利用多核处理器，我们根据原始流程中各部分算法的特征分别对它们进行了并行优化，流程各算法的变化如图 2 所示. 我们对 BWA-MEM 算法做了数据并行优化，而对于流程其他算法，我们利用任务并行的方式对其进行改进.

4.1　Smith-Waterman 算法的数据并行

Smith-Waterman 算法是典型的计算密集型算法，用于查找两个字符列的最优匹配，其形式化定义过程如下：定义查询序列 $Q=q_1 q_2 \cdots q_m$，参考序列 $D=d_1 d_2 \cdots d_n$. 当 $q_i \neq d_j$ 时，得分 $W(q_i,d_j)\leq 0$；当 $q_i=d_j$ 时，$W(q_i,d_j)>0$. G_{init} 和 G_{ext} 分别表示新出现一个空位(gap)和在已有空位基础上再添加一个空位的代价. 算法依次更新两个辅助矩阵 $\boldsymbol{E}$、$\boldsymbol{F}$ 以及得分矩阵 $\boldsymbol{H}$，得到最终的得分矩阵. 这三个矩阵的递推公式如下：

$$E_{i,j}=\max\begin{cases}E_{i,j-1}-G_{\text{ext}}\\H_{i,j-1}-G_{\text{init}}\end{cases} \tag{1}$$

$$F_{i,j}=\max\begin{cases}F_{i-1,j}-G_{\text{ext}}\\H_{i-1,j}-G_{\text{init}}\end{cases} \tag{2}$$

$$H_{i,j}=\max\begin{cases}0\\E_{i,j}\\F_{i,j}\\H_{i-1,j-1}-W(q_i,d_j)\end{cases} \tag{3}$$

对于 Smith-Waterman 算法，由式(1)～(3)可知，矩阵 $\boldsymbol{E}$、$\boldsymbol{F}$、$\boldsymbol{H}$ 的数据依赖关系如图 3 所示，可以看出 $\boldsymbol{E}$ 矩阵的每个元素的计算依赖于前一列的元素，$\boldsymbol{F}$ 矩阵的每个元素依赖于前一行的元素，而 $\boldsymbol{H}$ 矩阵元素的计算依赖于前一行以及当前元素的 $\boldsymbol{E}$ 和 $\boldsymbol{F}$. 数据的依赖关系导致在现有计算模式下难以进行向量化，特别是对于依赖同一行前一个元素的矩阵 $\boldsymbol{E}$，必须对数据进行重排变换，减少数据的依赖关系；另一方面，由于现有向量指令对向量宽度的限制，我们需要对数据进行分块.

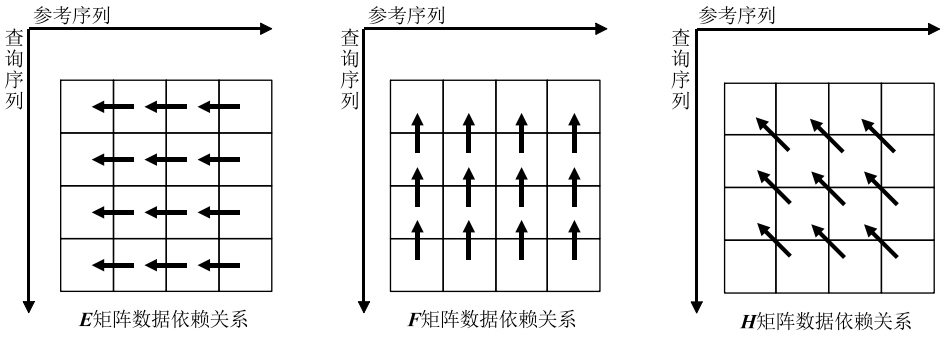


图 3 *E*、*F*、*H* 矩阵数据依赖关系图

变换方法如图 4 所示，通过对数据进行重排后，每一行由(1,2,3,…,12)变换为((1,4,7,10),(2,5,8,11),(3,6,9,12))三个向量块. 下一行的计算，依赖于上一行的前一个向量块值，以及同行的上一个向量块的值(对于下一行的第一个向量块，依赖上一行的最末尾的一个向量块，且注意到其相应位置有一个位置的偏移，计算前需要先对向量块进行移位操作)，此种方法解耦了同一个向量化块中的元素依赖. 具体到图 4 所示的例子，当计算第二行的第二个向量块(2,5,8,11)，其依赖的上一行的(2,5,8,11)和本行的(1,4,7,10)两个块在此时已计算完成并且这两块没有依赖关系. 对于边界上的块，如第二行的(1,4,7,10)，其依赖上一行的(1,4,7,10)和本行的(0,3,6,9)，事实上此处的“0”是不存在的，为了对齐数据进而便于向量运算，我们将本行的(3,6,9,12)右移一位变换为(0,3,6,9). 对于矩阵 *F* 和 *H*，它们依赖的上一行元素先于当前元素计算；但对于矩阵 *E*，它们依赖同行不同向两块中的元素，因此在第一遍处理时，同一行中会有多个元素的计算值不准确，但由于 *E* 的特殊限制条件，这些不准确的元素值大多会被忽略(计算 *H* 时根本用不到)，需要二次校正的值很少[6]，因此基于数据重排的向量化对于矩阵 *E* 的计算也带来较大的性能提升.

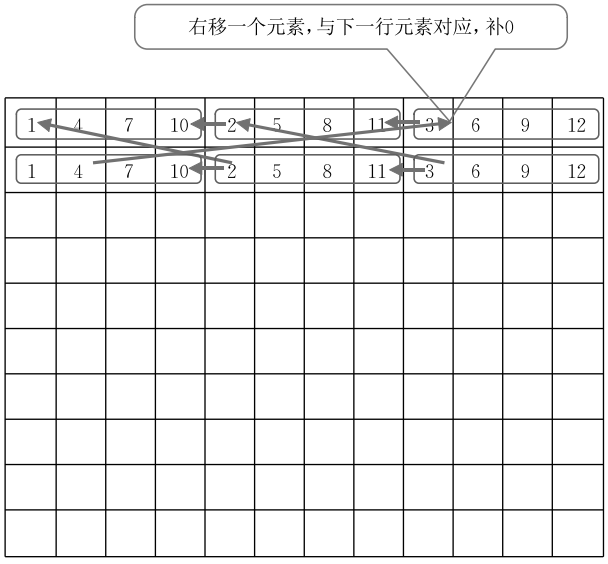


图 4 Smith-Waterman 算法数据重排

4.2 任务并行优化

Gene Panel 流程中 Smith-Waterman 后续的算法都非常适宜采用基于数据划分的任务并行，因为这类算法都以某种数据结构为单位进行处理. 它们的计算模式如图 5 所示，主要分为三步：任务划分、数据处理和结果合并. 对于一些临近数据有依赖的算法，可以通过添加一定长度的重叠扩展(overlap)来保证正确性. 比如，有的 read 跨越了任务划分的分界线，当前的划分边界无法将其完整的包含在任务内，我们可以将任务的原划分边界往外扩展一定的距离，将跨界的 read 整个包含进来，从而保证正确性. 需要注意的是，在实现过程中“任务划分”并非物理上将输入文件切分为多份文件，因为 BAM 文件(即输入文件)自带索引并且支持按区域(region)读取，因此我们只是逻辑上为每个线程分配 BAM 文件中特定区域的数据. 在多核共享存储系统中，只有一份 BAM 文件并且被多线程共享，每个线程按区域读取文件中相应的数据.

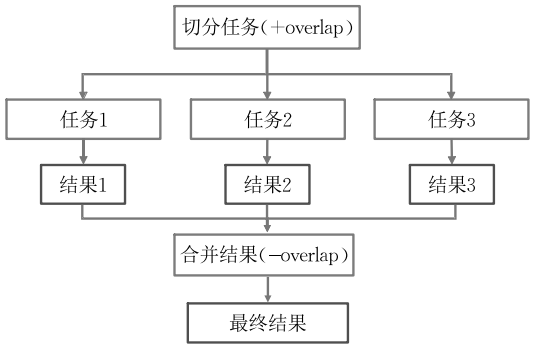


图 5 任务并行示意图

4.2.1 MarkDuplicate 算法的任务并行

MarkDuplicate 是检测 PCR 冗余的主要工具，

在讲述详细算法前，我们先介绍几个概念：

(1) Pair-end. 如果一条 read 和它的 mate read[①] 都被比对到参考序列上，则它们被称为一对构成 pair-end 的 read.

(2) Fragment. 如果互为 mate 的两条 read 其中一条 read 被比对上，而另一条 read 没有比对上，则这条比对上的 read 被称为 fragment(即同名的 read 配对成 pair-end，没有找到同名的表示一个 fragment).

(3) 判定冗余. 如果两个 pair-end 中对应的 read 的染色体(contig)，位置(position)和匹配的方向都相同，那么则说明这两个 pair-end 互为冗余；同样地，fragment 冗余的判定也类似. 此外，当一个 fragment 比对到染色体的位置和方向和一个 pair-end 相同，则优先选择 pair-end.

算法详细步骤如下：

1. 输入 BAM 文件(注意经过 sort 过程后的排序好的 BAM 文件)，内部利用一个缓冲区(buffer)作为辅助结构. 遍历文件，当发现两个 read 互为 mate，则将其打包成 pair-end，然后写入文件，遍历完成后 buffer 中剩下的都是未配对的 read，将他们都归为 fragment，并将它们写入文件.

2. 将 pair-end 和 fragment 按顺序排列，冗余的 pair-end 排序后处在相邻的位置，选择其中质量分数最高的 pair-end，其他的都标记为冗余，由于其有序性，只需要遍历一遍即可.

3. 对于 fragment，如果该 read 在 pair-end 中也存在，该 fragment 标记为冗余，否则选择质量分数最高的 fragment，其他的标记为冗余.

MarkDuplicate 算法中的主要依赖关系来自于构成 pair-end 的一对 read，构成 pair-end 的两个 read 在正常情形下是位置上相邻的. 大部分构成 pair-end 的两个 read 都会比对到原有位置上，如果一对 pair-end 比对的位置相距较远，原始流程会将它们视为错误比对而过滤掉. 因此对于两个相距较远的 read，我们认为它们不存在依赖关系，这与原始流程相符，不会引起误差. 因此，可以通过划分染色体区域来实现任务并行，将染色体区域分为等宽的区域，每个任务处理其中一个区域的数据. 如果构成 pair-end 的两个 read 被分别划分到相邻的两个任务，可以通过添加 overlap 来保证结果的正确性.

4.2.2　IndelRealigner 算法的任务并行

IndelRealigner 是以 interval 为单位进行处理的，它串行地处理上一步得到的 interval 文件. 我们沿用 MarkDuplicate 并行化方法中粗粒度的数据划分来实现任务并行，如图 6 所示，分布在两个不同线程的跨相邻 interval 的 read 会带来误差，由于其划分是基于 interval 的，因此添加 overlap 也无法避免误差，但由于这种 read 占总数的比例很小，因此误差几乎可以忽略.

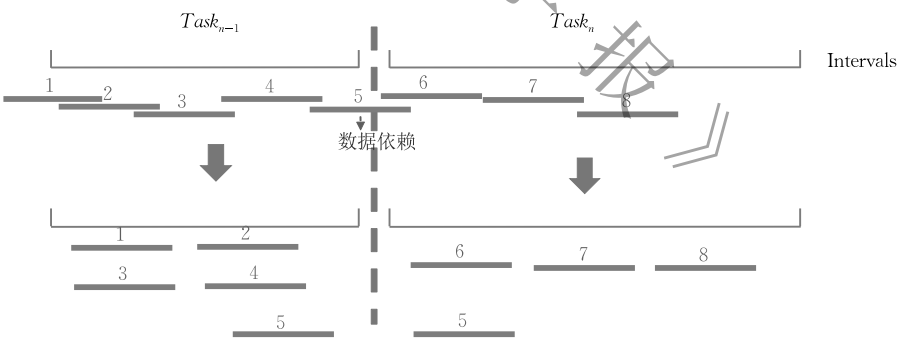


图 6　并行 IndelRealigner 算法

具体方法是将 RealignerTargetCreator 产生的 interval 文件(假设有 N 个 interval)根据线程数(假设有 T 个线程)均匀划分：$[0, N/T]$分配给第 1 个线程，$[2N/T, 3N/T]$分配给第 2 个线程等等，如此下去. 这种划分方式只会影响 T 个区间的数据(每两个线程交界处的 interval)，而 $N \gg T$，最后将每个线程的结果文件合并起来得到最终输出. 实际并行版本的结果与串行版本的结果差距非常小. 此外，每个 interval 的大小相差不大，基于这种划分方式，任务负载相对均衡，不需额外地处理.

4.2.3　Mpileup 算法的任务并行

Mpileup 算法中，将 Mpileup 子算法生成的位点信息通过管道导入到 Bcftools Call 子算法中. 算法的缺点主要有两个方面：首先算法是串行实现，无法有效利用多核计算资源；其次，两个子算法在进行变异位点发现过程中，由于参考序列上位点大概有 30 亿个，即使每个位点 10 字节数据，大概有 30 GB($30\times10^8\times10/10^9=30$ GB)的数据，管道读写数

① https://www.illumina.com/documents/products/technotes/technote_nextera_matepair_data_processing.pdf

据量大.针对该算法,一方面我们采用数据划分方式对其实施任务并行;另一方面,合并两个子算法,减少管道操作的开销.

Mpileup 算法以 pileup 为单位进行处理,其仍然可以沿用前面的基于数据划分的任务并行方式的再结合 overlap 来实现任务并行.如图 7 所示,每个线程在染色体上划分长度为 *CHUNK_SIZE*(一个定值,参数指定)的区间,然后将区间封装成一个任务后执行,其内部维护一个全局的同步信息.对于分跨两个区间的 read,采用添加 overlap 的方式来保证结果一致,overlap 默认取值为 150 bp(可作为参数调整).

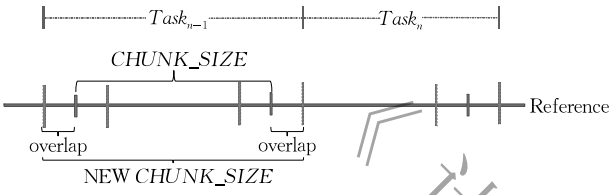


图 7 并行 Mpileup 算法

4.2.4 Mutect 算法的任务并行

原始 Mutect 算法基于 GATK 框架,只能使用串行的线性调度器,而且内部生成统计量过程中会引入很多冗余操作.

本文设计并实现了接口友好的高性能库 HCC(详见 5.3 节),并采用 HCC 重构 Mutect 的底层框架,同时优化内部操作.

与 IndelRealigner 划分方法类似,Mutect 的任务划分基于 panel 文件,但不同的是,Mutect 并行处理过程中 panel 之间是相互独立的,因此 Mutect 的并行处理不会引入误差.

5 辅助优化方法设计

5.1 冗余实现 Smith-Waterman 向量化

Smith-Waterman 算法主要由四个函数构成:*ksw_extend2*、*ksw_global2*、*ksw_i16* 和 *ksw_u8*,其中 *ksw_extend2* 和 *ksw_global2* 在现有实现中未使用 SIMD 优化,而在 *ksw_i16* 和 *ksw_u8* 已使用了 SSE2(采用 intrinsic 向量化指令)进行 SIMD 加速.本文对于 *ksw_u8* 和 *ksw_i16*,基于原有向量化算法,再为其添加 AVX2 支持.

ksw_extend2 和 *ksw_global2* 两个函数都是基于条带(Banding)的比对,如算法 1 所示.内层循环的 j 的迭代范围为$[i-w,i+w]$,w 为条带初始宽度(可指定,默认值为 100),同时每次计算完一行后,其会根据 $\boldsymbol{E}$ 矩阵和 $\boldsymbol{H}$ 矩阵的值来调整相应的 j 的范围(也就是改变 w 的大小).条带的好处是对于较长的 read,能够有效缩小比对的范围和减少计算量.

算法 1. 基于 Banding 的 Smith-Waterman 算法.

```
1.  beg←0
2.  end←0
3.  FOR i←1 to m
4.    beg←max(i−w,beg)
5.    end←min(i+w,end)
6.    FOR j←beg to end
7.      calculate E,F,H //Matrix operations
8.    ENDFOR
9.    WHILE E[i][beg]!=0 and H[i][beg]!=0
                                  //Update beg
10.     beg++
11.   ENDWHILE
12.   WHILE E[i][end]!=0 and H[i][end]!=0
                                  //Update end
13.     end−−
14.   ENDWHILE
15. ENDFOR
```

在算法 1 中,$[beg,end]$的区间大小不断缩小,导致没有一个固定的计算模式,无法很好地利用向量化指令.本文的采用一种增加冗余计算的方式,使得代码能够使用向量化方法进行优化加速.伪代码如算法 2 所示.

算法 2. 基于冗余的 Smith-Waterman 算法.

```
1.  beg←0
2.  end←0
3.  FOR i←1 to m
4.    beg←max(i−w,beg)
5.    end←min(i+w,end)
6.    FOR j←1 to n
7.      calculate E,F,H //Matrix operations
8.    ENDFOR
9.    //Clear the boundary to zero
10.   E[i][0..beg,end..n]←0
11.   F[i][0..beg,end..n]←0
12.   H[i][0..beg,end..n]←0
13. ENDFOR
```

将 j 的迭代范围扩展为$[1,n]$,对于在$[i-w,i+w]$区域外的数据在计算后对其置 0,如图 8 所示.这种方式对于长 read,如 1000 bp 以上,可能冗余的计算的 overhead 大于使用 SIMD 的提升,但是对于目前来说(大约 100 bp 左右),其能够有效加速性能.

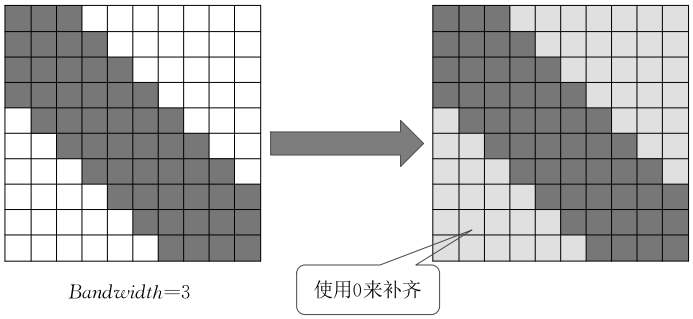


图 8　基于冗余的 Smith-Waterman 算法示意图

上述向量化方法会引入冗余计算，会减少向量化的收益，本文提出一种量化选择方法，根据 read 长度以及 band 宽度等参数来权衡向量化方法带来的性能提升以及冗余开销，从而决策是否应采用向量化方法.

假设 a 表示向量化加速比，w 表示 band 宽度，n 表示查询序列的长度，则有如下公式：

$$\frac{n}{a} \leq 2w \tag{4}$$

不等式(4)中，n 为使用向量化优化后的计算量，而 $2w$ 为原始实现的计算量. 当不等式成立时，采用向量化方法的收益大于冗余计算带来的开销，此时应选择向量化实现方法；否则，当不等式不成立，应放弃向量化优化手段. 此外，一般 a 的值略小于向量化寄存器包含的元素个数，可根据实验来选定 a 的值.

5.2　负载均衡

对于 Gene Panel 流程，待处理的 read 在染色体组上分布是不均匀的，read 主要集中于 panel 覆盖的染色体区域. 当通过均分染色体区域来划分任务时，有些任务处理的染色体区域包含较多的 panel 区间，而有些任务分配的区域基本没有 panel 区间覆盖，这就导致不同任务处理的 read 数目可能差距很大，从而导致严重的任务划分不均匀而影响并行处理性能. 但是我们知道，Gene Panel 流程输入的 read 集中分布在 panel 文件所覆盖的区间，于是负载均衡就可以通过均衡划分 panel 区间来实现.

为此，我们设计了一种任务均衡划分算法. 假设处理单元(计算核心)数目为 n，待处理的染色体区间为$[1,r]$，算法步骤如下.

算法 3.　负载均衡的并行算法.

S1. 先对 panel 文件进行预处理，将有 overlap 的区间合并，并删除无效区间，得到 m 个互不连续的顺序区间：

$$[l_1,r_1],[l_2,r_2],[l_3,r_3],\cdots,[l_m,r_m],$$

其中 $l<l_1<r_1<l_2<r_2<\cdots<l_m<r_m<r$.

S2. 将上述 m 个区间扩展为 m 个连续的染色体区域：

$$[l,r_1),[r_1,r_2),[r_2,r_3),\cdots,[r_m,r).$$

S3. 通过 BAI 文件(BAM 文件的索引文件)计算 S2 得到的每个区域包含的 read 数目；

S4. 对 m 个区域按其包含的 read 数目从大到小排序得到：$a_1,a_2,\cdots,a_m$.

S5. T_i 表示处理单元 P_i 的任务集，首先依次将 $a_1,a_2,\cdots,a_n$ 分配给 $T_1,T_2,\cdots,T_n$；之后，根据“贪心算法”的思想，按 从大到小的顺序依次分配 $a_{n+1},a_{n+2},\cdots,a_m$，每次接收任务的是当前任务集内 read 总数最小的处理单元.

S6. 任务划分完成后，所有处理单元并行执行相应的算法，每个处理单元处理其任务集内的任务.

S7. 按任务划分顺序合并结果文件.

5.3　高效的基因分析底层库设计——HCC

目前很多现有基因序列分析工具，典型的如 GATK，其依赖底层的 HTSJDK 库以及其他使用到的 JAVA 库，它们相对于以 C 开发的 HTSLIB 库性能较差，但是 HTSLIB 其本身代码结构以及各种 API 的学习和使用成本过高，对于一些使用 C 或 C++ 作为开发语言的开发者而言没有一个易用而且高性能的基因序列分析底层库. 本文基于 HTSLIB，设计并实现了高性能基因序列分析库 HCC，一方面对原有 HTSLIB 结构进行封装，另一方面提供一些常用的数据分析处理工具类.

HCC 的主要目的是为上层基因分析工具的开发提供简单、易用、高性能的 API，如图 9 所示，HCC 整体分为四个部分：最底层的 HTSLIB，库中大部分基因分析类都是基于 HTSLIB 原始结构建立；

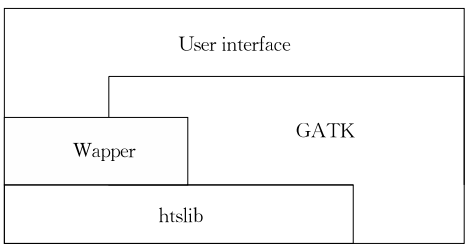


图 9　HCC 库设计框架图

Wrapper 对 HTSLIB 的一些原始结构的封装，同时也基于 HTSLIB 提供一些工具类，如对 bam1_t 的封装类 SAMBAM-Record 等；GATK 层是在 Wrapper 和 HTSLIB 的基础上，构建一些 GATK 框架特有的数据结构（GATK 某些功能实现与 HTSLIB 有差异，为了同时支持基于 HTSLIB 开发的工具和 GATK 工具），如 *GATKPileupTraverse*、*AlignmentStateMachine* 等；最上层是面向用户的 API，用户可以用这些 API 来构建基因分析工具.

HCC 利用面向对象的方法封装原始的 HTSLIB 对象，提供给用户统一的操作接口，图 10 展示了 HCC 库中主要类的关系结构，下面简要介绍几个主要类的作用：

(1) *FastaIndex* 封装 *faidx_t* 结构，提供创建、加载和销毁 Fasta 索引文件等功能.

(2) *AbstractFastaSequenceFile* 是一个抽象类，提供一些读写 Fasta 文件的基本方法.

(3) *VCFHeader* 是对 *bcf_hdr_t* 结构的封装，是操作 VCF 头部信息的类.

(4) *VariantContext* 是对 *bcf1_t* 的封装，提供操作 VCF 记录的功能.

(5) *VCFReader* 是 *VCFTextReader* 和 *VCFIndexReader* 的基类，提供读取 VCF 文件的基本接口.

(6) *VariantContextWriter* 是 *VCFWriter* 和 *SortingVariantContextWriter* 的基类，其中 *VCFWriter* 提供写入变异位点记录的基本功能，*SortingVariantContextWriter* 提供对 VCF 文件按照位置有序写入的功能.

(7) *SAMBAMRecord* 是对 *bam1_t* 的封装，包含操作基本 BAM read 的常用方法，如获取 read 的位置等.

(8) *AbstractSAMBAMReader* 是 *SAMBAMTextReader*、*BAMIndexReader* 以及 *BAMIndexBatchReader* 的基类，其中 *SAMBAMTextReader* 提供按行顺序读取 BAM/SAM 文件的功能，*BAMIndexReader* 和 *BAMIndexBatchReader* 提供随机读取 BAM 文件的功能.

(9) *BAMWriter* 提供写入 BAM 文件的功能.

(10) *PileupElement* 是对 *bam_pileup1_t* 的封装，表示 Pileup 的一个元素.

(11) *PileupTraverse* 是对 HTSLIB 中获取 Pileup 的算法的封装，*GATKPileupTraverse* 是对 GATK 中获取 Pileup 的算法的封装，这样可以同时支持依赖于 HTSLIB 和 GATK 的工具.

(12) *LevelingDownsampler*、*DiploidGenotye*、

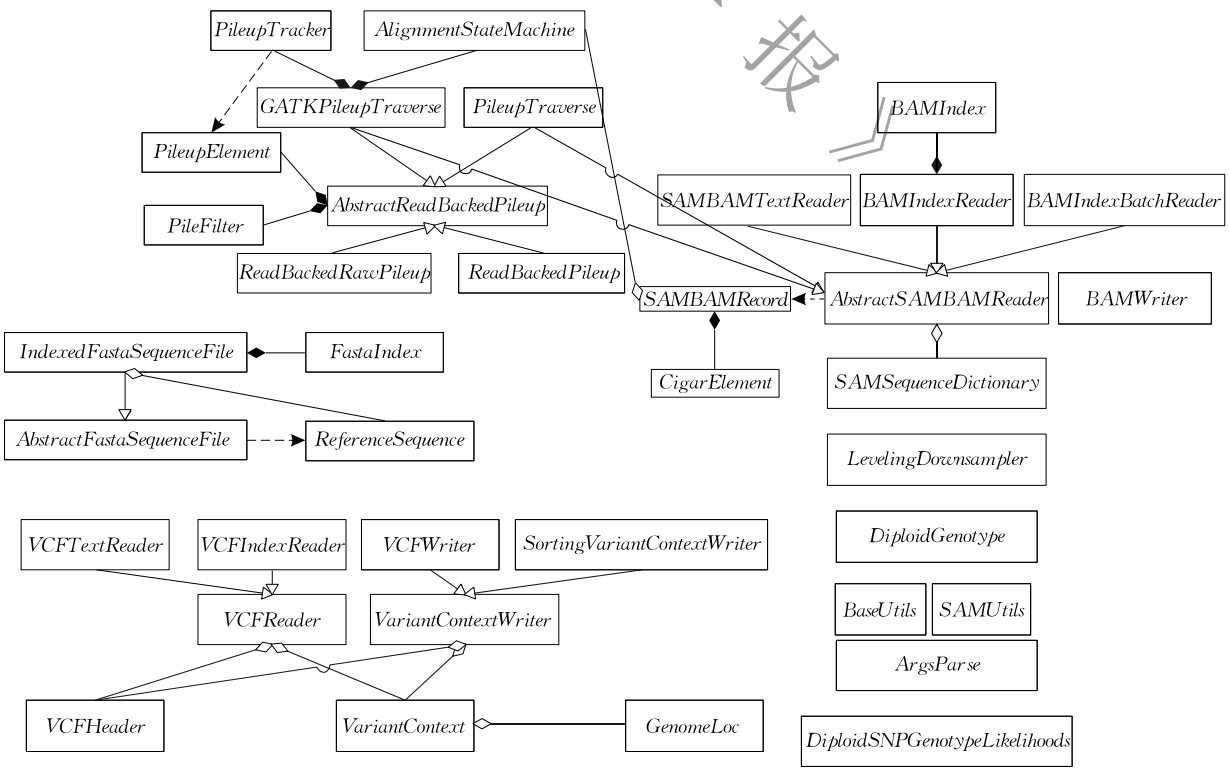


图 10 HCC 库中主要类的关系结构图

BaseUtils、*BAMUtils* 等是工具类.

上面这些类主要分为三种，第一种是对 HTSLIB 的原始数据结构进行封装，第二种是对 GATK 中区别于原始 HTSLIB 实现的算法的实现和封装，第三种提供一些工具类.

5.4　算法间优化

上面叙述的优化手段都是针对单个算法，本节我们讨论在算法与算法之间的执行顺序和中间结果上实施的两种优化手段，分别是“算法重叠执行”和“算法间数据缓存”，如图 11 所示.

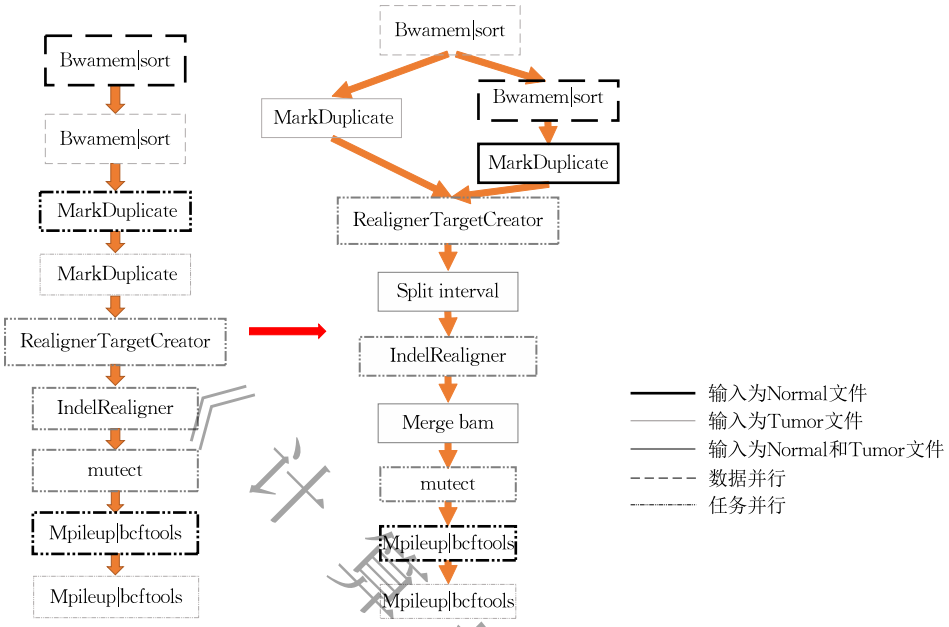


图 11　流程算法间优化示意图

5.4.1　算法重叠执行

在 Sample 预处理阶段，由于 MarkDuplicate (IO 密集型)会划分/合并输入/输出文件，而这部分 IO 都是较大的 BAM 文件，严重阻塞了流程的执行；另外，Tumor 和 Normal 数据的处理互相不依赖，故可以同时处理两个 Sample 数据，从而保证计算(计算密集的 BWA-MEM)和 IO(IO 密集的 MarkDuplicate)操作并行执行.

5.4.2　算法间数据缓存

流程处理过程中，会产生很多中间文件，如 BWA-MEM 进行数据比对后会生成 BAM 文件，MarkDuplicate 和 IndelRealigner 也会生成处理后的 BAM 文件，这部分中间文件读写会耗费一定时间. Tmpfs① 是 Linux/Unix 系统上的一种基于内存的文件系统，tmpfs 可以使用内存或 swap 分区来存储文件，因此速度较快. 本文使用 tmpfs 内存文件系统来存储中间计算数据，显著减少对磁盘的访问，提升中间计算 IO 的性能.

6　设计评测

在前面章节中，本文分别对 Gene Panel 流程中各种工具的算法和优化设计进行了详细的阐述，本章将对这些优化设计方法的效果进行全面细致的测试和分析. 本章首先介绍实验平台的配置和数据的选取，然后对各种优化手段的效果进行实验评测及分析.

6.1　性能评估方法

实验平台. 硬件平台配置如表 2 所示，我们采用主频 2.9 GHz 的双路 Intel Xeon E5-2680 CPU，每个 CPU 有 12 个核，开启超线程情况下每个 CPU 可运行 24 个线程；内存采用 256 GB 的 4 通道 DDR4 内存，总带宽为 68 GB/s.

表 2　实验平台硬件系统

CPU	2×12 Intel Haswell Xeon Cores@2.5 GHz
DRAM	4×DDR4-2133 17 GB/s channel (68 GB/s)

Gene Panel 流程. 流程所用到的各算法实现的选择如表 3 所示，BWA 工具包提供了 BWA-MEM 等比对算法的实现；Samtools 工具包提供了 Mpileup 等算法实现；GATK 工具包提供了 IndelRealigner 等算法实现.

① https://wiki.archlinux.org/index.php/tmpfs

表 3 算法配置

软件	版本
BWA	0.7.15-r1142
samtools	1.3.1-34-g26e1ea5
HTSLIB	1.3.1-35-g481752c
MarkDuplicate	1.119
GATK	3.3-0-g37228af
Mutect	1.1.7

参照序列及样本数据. 人类参照序列等共用数据库如表 4 所示,其中人类基因参照序列采用 1000 Genome① b37.样本数据如表 5 所示,本文选用的 3 份样本数据,其中 MG225 和 MG271 是 Gene Panel 数据,另一份是 WES 数据,它们都是基于某研究所提供的病人的基因数据.

表 4 共用数据库

类型	参考数据源	组织	大小
Reference	1000 Genome b37	Homo sapiens	3G
dbsnp	dbsnp_137	Homo sapiens	10G
cosmic	cosmic_137	Homo sapiens	972K
indel_1000G	1000G phase1	Homo sapiens	227M
indel_mills	Mills&-1000G Gold standard	Homo sapiens	83M

表 5 样本数据集

数据类型	MG225	MG271	WES
Tumor	7.4G	18G	25G
Normal	2.4G	2.1G	7.4G
Panel	mutect_panel_372(485K)	mutect_panel_372(485K)	SureSelect.Human.All.Exon.V5.intervals(8.5M)

评价指标及测试方法. 为评价本文对 Gene Panel 流程所做的并行化和其他优化的效果,我们关注:(1)性能;(2)多核 CPU 资源利用率;(3)正确性以及(4)可扩展性四个指标.我们将原始 Gene Panel 流程作为基准,用原始流程与优化后流程在实验平台上的执行时间之比作为加速比,用以衡量性能提升;论文通过在实验平台部署 nmon② 工具来测量多核 CPU 资源利用率,优化前后作为比较;对于正确性,我们比较优化后流程与原始流程最终输出的变异位点"raw variants",找出二者重合与有差别的"变异位点",误差率表示为有差别变异位点数与原始流程变异位点数之比,误差率反映优化后流程的正确性;最后,我们分别测试流程中各个算法并行化之后在不同线程数目配置下的执行时间,来反映并行算法的可扩展性.

6.2 评测结果及分析

6.2.1 总体效果

图 12 展示了优化后的 Gene Panel 流程在 48 线程下的加速效果.基准(Baseline)是原始的 Gene Panel 流程的性能,后面依次是流程各个算法(步骤)并行和优化后带来的整体加速效果.在所有算法完全并行优化之后,数据集 MG225 和 MG271 分别有 5.08 倍和 3.73 倍的整体加速;如果将 BWA 独立出去(严格意义上,BWA 不属于 Gene Panel 流程),整个流程的加速比在两个数据集上分别提升到 7.6 倍和 6.4 倍.对于 MG271 的流程的加速比不高主要原因是由于 MG271 的 tumor 数据更大,BWA-MEM 耗时较多,而由于 read 长度选择的限制,BWA-MEM 的加速比不高,导致 MG271 下的整体流程的加速比较 MG225 更低.

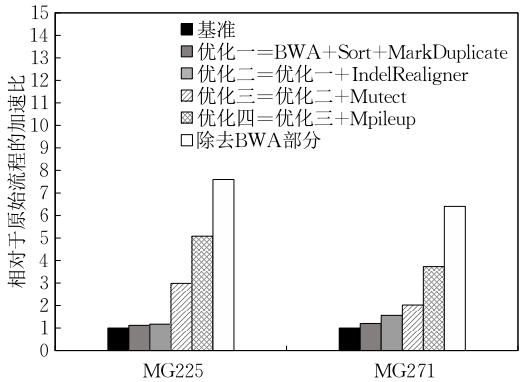


图 12 Gene Panel 各算法的并行与优化加速效果

从图 12 可以看出,整体加速主要是由 Mutect 和 Mpileup 算法的并行优化带来的,这是由于:(1)这两个算法都是多线程实现,不存在划分任务与合并文件的开销;(2)这两个算法执行时间占到整个流程的将近一半,因此,他们对整体性能影响很大.但是,另一个时间占比较大的算法 BWA-MEM 由于实际 read 长度的限制,其 SIMD 优化带来的收益有限,因此它严重影响了整体性能的提升,但好在随着测序技术的发展以及测序仪的改进,read 将越来越长,此优化的效果也将会显著提升.而相比于 Mutect 和 Mpileup,前面的几个算法由于并行开销严重(MB 至 GB 量级的任务划分与文件合并),它们的性能提升相对而言也就没那么明显.

6.2.2 CPU 资源利用率

数据集 MG225 优化后的流程资源利用率如图 13 所示,可以看出大部分时间资源利用率都很高,但是也可以看出其中有一部分时间段资源利用率不高.0～55 这个时间段主要是 BWA-MEM 进行比对,其中 38～55 这段时间主要是 BWA-MEM 处理最后一部分 read,同时将比对好的 BAM read 写

① https://www.genome.gov/27528684/1000-genomes-project/
② http://nmon.sourceforge.net/pmwiki.php

入管道文件，这主要是 IO 操作，其中 37～45 主要是 BWA-MEM 将数据写入管道，51～55 主要是将排序好的数据写入文件中，而在 46～50 资源使用率达到 40%左右，这是由于管道中数据填充满后触发了 sort 过程. 时间段 150～172 主要是将 IndelRealigner 后的 BAM 文件合并到一起，这部分主要是 IO 操作. 中间有些资源利用率下降主要由于其中某个工具处理完数据然后将数据写入文件中，其他时间段的资源利用率都基本达满负荷运行.

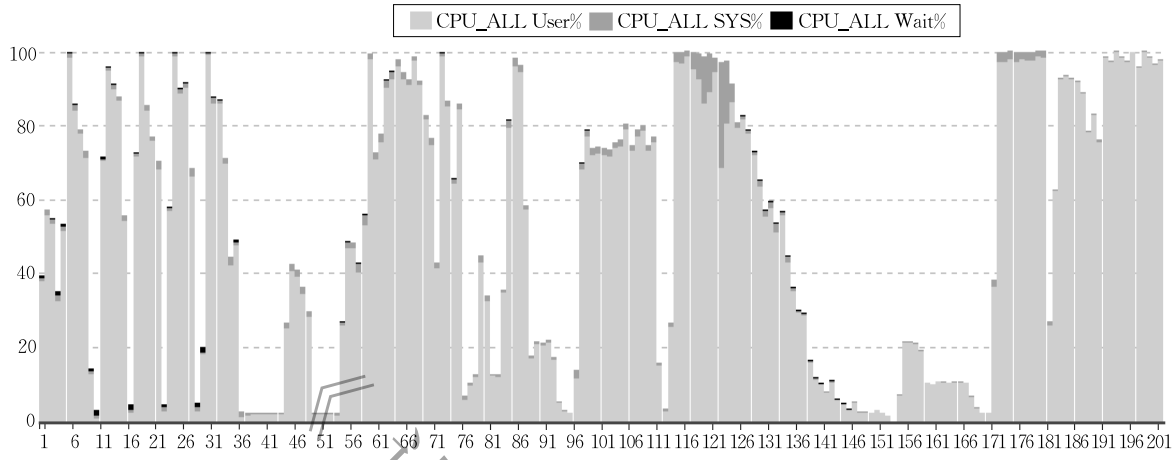


图 13　并行化流程的 CPU 资源利用率

6.2.3　正确性

我们以数据集 MG225 为例，将本文优化版本的“raw variants”结果与原始流程产生的“raw variants”进行比较，结果如图 14 所示.

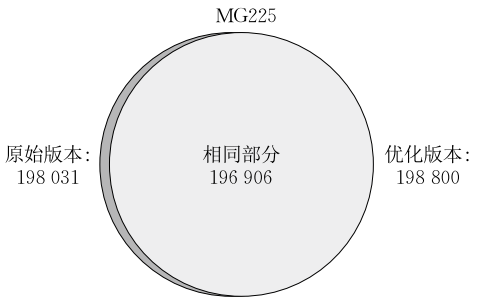


图 14　并行化流程的误差

从图 14 中可以看出，原始流程共找出了样本中 198031 个变异位点，而优化流程召回了 198800 个，其中有 196 906 个与单机流程找出的变异位点重合. 基因分析流程中的变异位点分析工具（如 Mutect、Mpileup）本身就是基于统计学概率的算法，算法本身就包含不确定性，就其本身而言，谈绝对准确是没有意义的[①②]. 测试结果显示，经过并行优化后的流程基本保持了和原始流程相同的结果.

6.2.4　基于冗余的 Smith-Waterman 向量化

测试 BWA-MEM 中 kernel 性能，本文选用的参考序列数据为人类基因组 8 号染色体数据，不同长度的 read 数据是使用 Wgsim[③] 生成的.

本文测试三个优化后的函数在不同 read 长度下的加速比，如图 15 所示. 函数 *ksw_align* 加速比是优化版本（AVX2）相对于原始版本（SSE2）的加速比，平均在 1.5 倍左右；AVX2 向量长度为 SSE2 的 2 倍，但没有达到 2 倍的加速比，原因在于 AVX2 指令开销比 SSE2 指令的开销大. 优化后的 *ksw_extend2* 函数平均也有 1.5 倍的加速. 而对于函数 *ksw_global2*，其使用 SIMD 后性能不增反降，主要由于其 band 宽度太小，SIMD 的 padding 方式带来的开销大于其收益；同时，里面含有很多位操作，这也增大了 SIMD 实

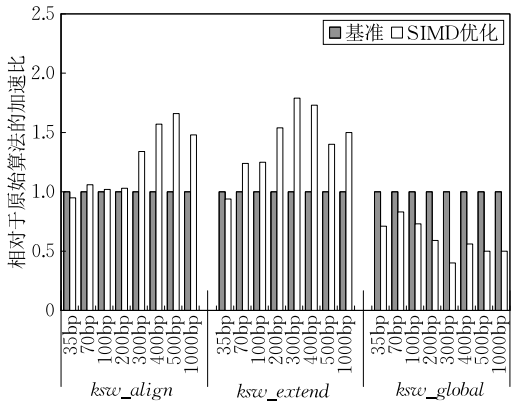


图 15　Smith-Waterman 中各函数 SIMD 优化效果

① https://blog.genohub.com/2016/10/24/targeted-gene-panels-vs-whole-exome-sequencing/
② https://software.broadinstitute.org/gatk/documentation/article?id=11036
③ https://github.com/samtools/samtools

现的开销.

从图 15 还可以看出,Smith-Waterman 算法的 SIMD 优化在 100～1000 bp 时效果明显,也就是说 read 越长加速越明显,而随着测序技术的发展以及测序仪的改进,read 长度越来越长,因此,优化的意义会逐步显现.不过,在 read 长度达到一定程度后,性能开始下降,这是由于 SIMD 的冗余开销越来越大.

上述性能测试结果只针对单个核心函数,为了衡量各个核心函数在整个程序中发挥的作用,我们又测试了几种核心函数通过数据并行优化分别带来的整体性能提升,如图 16 所示.图 16 中 *bwa* 表示 BWA 工具的原始实现,*bwa_align* 表示原始实现中单独对 *ksw_align* 函数使用 AVX2 指令加速,*bwa_extend* 表示原始实现中单独优化 *ksw_extend*,*bwa_global* 表示原始实现中单独对 *ksw_global* 函数进行 SIMD 优化.

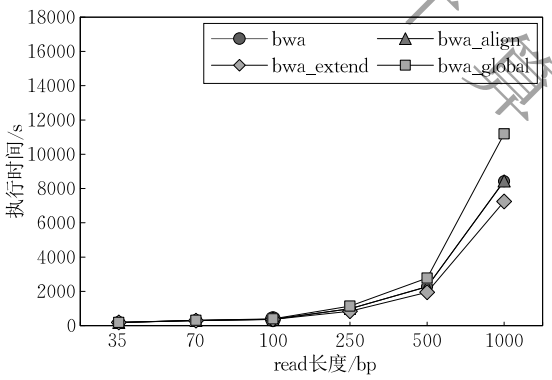


图 16 Smith-Waterman 中各函数单独优化对 BWA 整体执行时间的影响

从图 16 中可以看出,在 read 长度大于 100 bp 后,单独优化几种核心函数都会带来整体性能的提升;而在 read 长度大于 250 bp 后,加速效果能达到 10%～25%.另一方面,三个核心函数中 *ksw_extend* 带来的加速效果最为明显,原因是其在整个程序中时间占比较大;相比而言 *ksw_align* 带来的加速效果有限,这是由于 *ksw_align* 函数只会在一对构成 *pair-end* 的两条 read 比对位置差距过大需要重新比对时才会被用到,而这部分在整个程序中时间占比较少.

如图 17 所示,我们综合上述三种核心函数的优化,使得 BWA-MEM 的整体性能有平均 1.1 倍的加速比;另一方面,是否开启多线程对 Smith-Waterman 向量化加速效果基本无影响.

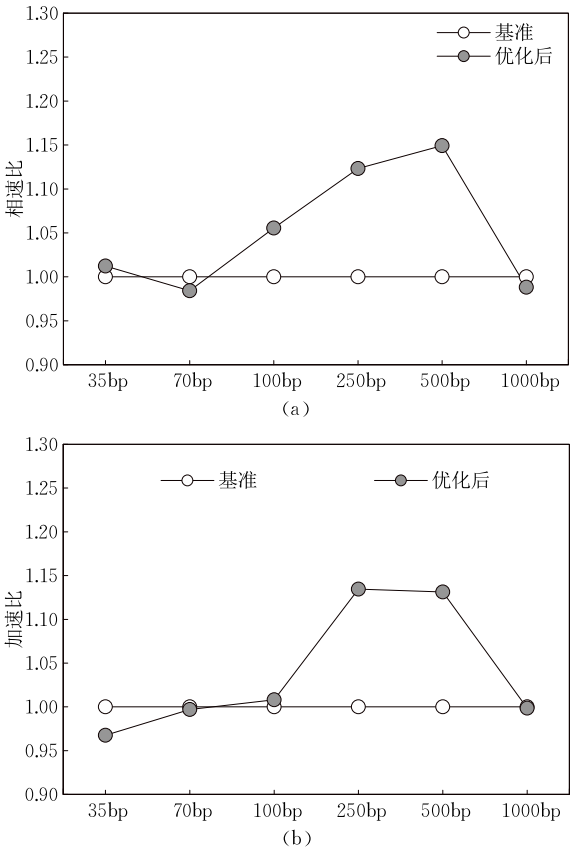


图 17 Smith-Waterman 整体优化在(a)单线程和(b)24 线程下相对 BWA 整体的加速比

6.2.5 MarkDuplicate 性能评测

本文选用 MG225-normal 数据集来测试 MarkDuplicate 的性能.如图 18 所示,MarkDuplicate 在 24 线程配置下相对单线程有 5 倍加速比.在 MarkDuplicate 算法中,任务并行的主要开销来源于任务划分以及结果合并,这部分工作只能是串行执行的,并且其在算法执行时间中占有一定比例,因此加速效率会受到限制.

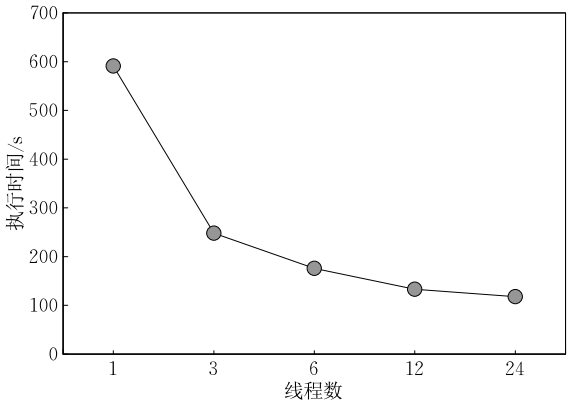


图 18 MarkDuplicate 在不同线程下的扩展性

6.2.6　IndelRealigner 性能评测

本文选取 MG225-narmal 数据集，如图 19 所示，IndelRealigner 在 24 线程配置下相对于单线程可获的 8 倍的加速比. 同样的，IndelRealigner 需要划分任务并合并结果文件，这部分必须串行执行，这部分开销导致加速比无法呈线性.

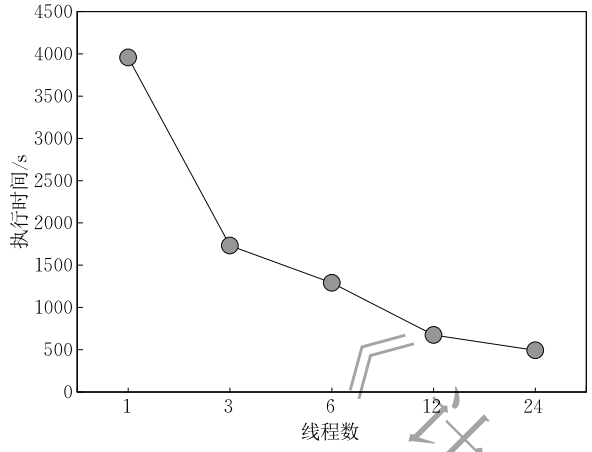


图 19　IndelRealigner 在不同线程下的扩展性

6.2.7　Mpileup 性能评测

对于 Mpileup 算法，*CHUNK_SIZE* 是一个重要参数，如果其取值过大，有可能造成任务分配不均匀，导致部分任务拖慢整个处理过程；而如果其取值过小，则可能导致任务的碎片化而增大开销. 我们在性能测试实验之前先对 *CHUNK_SIZE* 进行调优，本文通过采样的方法（如每条染色体取 1% 的数据）测试其性能，如图 20 所示，性能会随着 *CHUNK_SIZE* 的增大先下降后上升. 因此本文将 *CHUNK_SIZE* 值设为 500 000 bp.

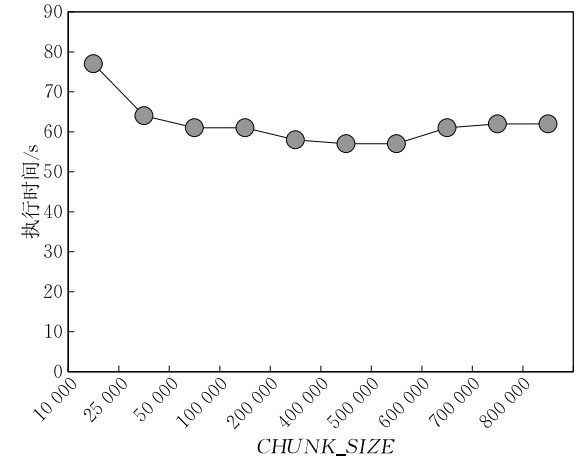


图 20　参数 *CHUNK_SIZE* 对 Mpileup 运行时间的影响

根据 5.2 节算法分析可知，read 的深度在染色体上分布不均匀可能导致“长尾”任务，拖慢整体处理速度，本文对 Mpileup 算法进行任务并行优化，并结合了任务均衡划分，效果如图 21 所示. 我们通过对 Mpileup 原始算法代码层面的优化，在没有并行化的情形下（单线程）也获得了 1.3 倍的加速比. 单线程下负载均衡也有加速的原因是我们的负载均衡算法首先会对区间文件进行预处理，合并重叠区间并删除无效区间，这就减少了后续遍历各区间上位点的工作量. 总体来说，最优化版本（多任务且任务均衡）在 48 线程配置下相比原始版本有 32 倍加速比；在 48 线程的配置下，任务均衡相对于仅采用多任务而未进行任务均衡的版本获得了 1.6 倍的加速比.

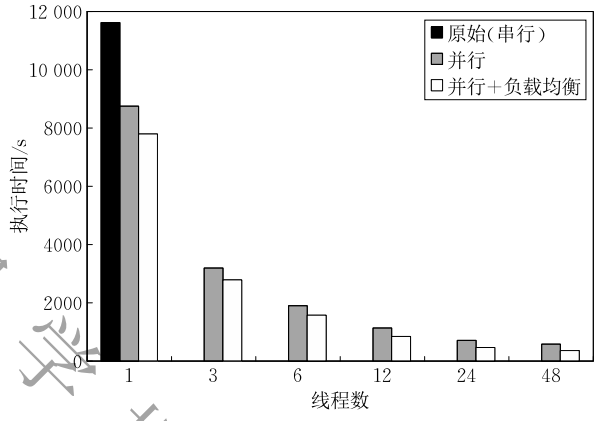
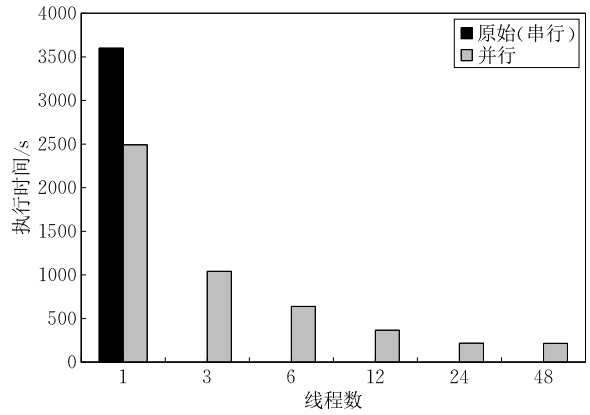


图 21　Mpileup 在不同线程下的扩展性

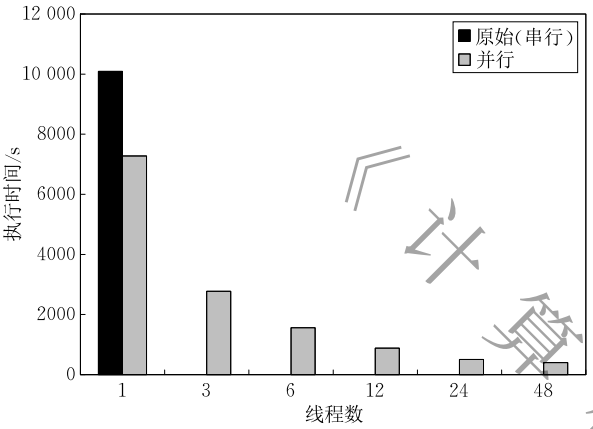
6.2.8　Mutect 性能评测

Mutect 作为变异位点发现的工具，可用于 Gene Panel 流程中，也可用于 WES 流程中，但由于两个流程的数据扩增倍数不同，可能导致程序运行特征也不相同，为了测试本文对 Mutect 的优化方法的通用性，本文除了选用 Gene Panel 的 MG225 数据集，还增加了 Mutect 对 WES 数据集的性能测试.

从图 22(a)和图 22(b)可以看出，我们通过对 Mutect 原始算法代码层面的优化，在没有并行化的情形下（单线程）也获得了 1.4 倍的加速比. Mutect 在 MG225 数据集下，48 线程相对于单线程可获得 16.8 倍的加速比，而在 WES 数据集下加速比能达到 25.3 倍. 这不仅说明我们的优化方法适用于 Gene Panel，在 WES 流程中也可以获得较好的加速效果.



(a) Gene Panel



(b) WES

图 22 Mutect 在不同线程下的扩展性

6.2.9 负载均衡效果评测

以数据集 MG225 为例,MarkDuplicate 算法的负载均衡的加速效果如图 23 所示.从图中可以看出,MarkDuplicate 的数据划分均衡后会带来平均 1.4 倍的性能提升.从图中还可以得到如下结论:

(1)在划分数目为 48 时,均分和负载均衡划分性能差距不大,主要原因是 Tumor(2.3 G)和 Normal(465 M)划分成 48 份后每份的任务都很小,即使有数据不均衡,其影响也不大,划分的份数越少,负载均衡的效果就体现越明显.

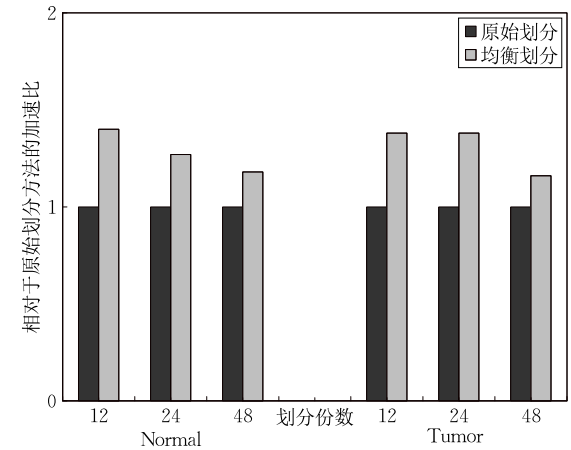


图 23 负载均衡优化加速比

(2)负载均衡方式的合并时间比均匀划分方式的合并时间短,主要原因是负载均衡的任务产生的结果也更均衡,文件差距越小,合并的速度越快.

(3)划分份数为 12 时为最佳,虽然 24 份相对于 12 份的执行时间减少,但是合并时间增大,相比而言 12 份时的总耗时更小.

6.2.10 HCC 效果

本文以如下一个案例来展示 HCC 的使用,统计 BAM 文件中的位点的平均深度,基于 HCC 有如算法 4 代码,其中 *PileupTraverse* 封装了遍历染色体位点的 pileup 算法,*fun* 为回调函数,在每次读取一行 read 时调用,可以在其中插入过滤函数,如过滤质量分数低的 read.

算法 4. 统计 BAM 文件中的位点的平均深度.

```
1.  BAMIndexReader reader(fp);
2.  TraverseData data;
3.  data.reader=&reader;
4.  PileupTraverse traverse(fun,&data);
5.  double total_depth=0;
6.  size_t site_cnt=0;
7.  while (traverse.HasNext()) {
8.      auto&&pileup=traverse.Next();
9.      total_depth+=pileup.Size();
10.     site_cnt+=1;
11. }
12. printf("位点的平均深度:%lf\n", total_depth/
    site_cnt);
```

上例只用了 10 行左右代码实现了该功能,而且其接口相对于 HTSLIB 来说,API 的定义和使用都很友好,能够极大提高编程人员的编程效率.

我们基于 HCC 开发的 mutect 的代码量大概 3000 行左右,相比于原始算法的 4000 行左右,工作量减小很多,且并行版本的 Mutect 的性能在单线程情况下相对原有基于 GATK 框架的版本有 1.4 倍的加速.

6.2.11 算法间优化效果

图 24 展示了算法间优化的加速效果,其中前四对柱形表示各组算法单独进行算法间优化后,与它们各自实施算法间优化之前相比获得的加速比;最后一对柱形表示所有算法都实施算法间优化后,整个流程相对没有进行算法间优化时的加速比.

由于计算密集的 BWA-MEM 与 IO 密集的 MarkDuplicate 的重叠执行,BWA-MEM 与 MarkDuplicete 重叠执行部分获得了 1.55 倍的加速比.

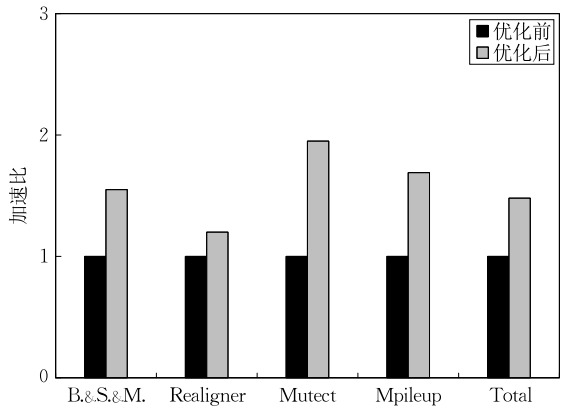


图 24　算法间优化加速比

而在后续算法中，通过采用内存文件系统，减少了每个步骤中间文件的读写时间，获得了最高 1.95 倍的加速. 单独的算法间优化又给整个流程带来了 1.48 倍的性能提升.

7　总结及下一步工作

为了应对爆炸式增长的基因测序大数据并进一步降低测序成本，提升基因数据分析流程的性能成为日益迫切的需求. 本文以 Gene Panel 流程为研究对象，设计并实现了一套全新的并行 Gene Panel 基因数据分析流程. 通过数据并行和任务并行两种主要手段，同时结合负载均衡等其他优化方法，有效地提升了多核平台的资源利用率，并获得了 4～7 倍的整体性能提升. 在不损失性能的前提下，本文还设计实现了一套接口简单易用的高性能基因数据分析底层算法库.

然而，本文工作还有进一步完善的空间：首先，在目前流程中，还是有大量地磁盘 IO 操作，非常耗费时间，未来可以将流程中所有工具合并为一体，采用基于流式的数据处理方式，减少工具之间的中间 IO 操作，从而进一步提升流程整体性能；其次，完善高性能基因分析库 HCC，集成一些 GATK 框架中常用的功能，提供一个高效、易用和更加完善的基因分析底层算法库是另一个重要的研究方向.

致　谢　感谢各位编辑和审稿人为本文的完善所提出的有益建议！

参　考　文　献

[1] Hall M J, et al. Gene panel testing for inherited cancer risk. Journal of the National Comprehensive Cancer Network, 2014, 12(9): 1339-1346

[2] Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv preprint arXiv: 2013, 1303.3997

[3] Langmead B, Salzberg S L. Fast gapped-read alignment with Bowtie2. Nature Methods, 2012, 9(4): 357-359

[4] Darling A, Carey L, Feng W. The design, implementation, and evaluation of mpiBLAST. Proceedings of ClusterWorld, 2003, 2003: 13-15

[5] Smith T F, Waterman M S. Identification of common molecular subsequences. Journal of Molecular Biology, 1981, 147(1): 195-197

[6] Farrar M. Striped Smith-Waterman speeds database searches six times over other SIMD implementations. Bioinformatics, 2007, 23(2): 156-161

[7] Klus P, Lam S, Lyberg D, et al. BarraCUDA—A fast short read sequence aligner using graphics processing units. BMC Research Notes, 2012, 5(1): 27

[8] Manavski S A, Valle G. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. BMC Bioinformatics, 2008, 9(2): S10

[9] Liu Y, Maskell D L, Schmidt B. CUDASW++: Optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units. BMC research notes, 2009, 2(1): 73

[10] Vouzis P D, Sahinidis N V. GPU-BLAST: Using graphics processors to accelerate protein sequence alignment. Bioinformatics, 2011, 27(2): 182-188

[11] Ren S, Sima V M, Al-Ars Z. FPGA acceleration of the pair-HMMs forward algorithm for DNA sequence analysis//Proceedings of the 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). Washington, USA, 2015: 1465-1470

[12] Kelly B J, Fitch J R, Hu Y, et al. Churchill: An ultra-fast, deterministic, highly scalable and balanced parallelization strategy for the discovery of human genetic variation in clinical and population-scale genomics. Genome biology, 2015, 16(1): 6

WANG Yuan-Rong, Ph. D. candidate. His major interests focus on software-hardware co-design on biological big data processing.

ZENG Ping, M. S. candidate. His research interests focus on parallel framework design on biological big data processing.

ZANG Da-Wei, Ph. D., assistant professor. His interests focus on computer architecture.

TAN Guang-Ming, Ph. D., professor, Ph. D. supervisor. His interests include high-performance computer architecture and parallel algorithm.

SUN Ning-Hui, Ph. D., professor, Ph. D. supervisor. His main research interests include computer architecture, high performance computing and distributed OS.

Background

Nowadays, the costs of genome sequencing are rapidly reduced, with the rapid development of the next generation sequencing (NGS) technology. However, the improvement of genome sequencing technology is resulting in gene data explosion at the same time. In fact, sequencing is just the first step in the clinical disease detection process. The sequenced gene data outputted from the sequencer need to be further processed by several analysis tools. The genetic data analysis tools are gradually unable to meet such a large-scale data analysis demand. On one hand, most of the genetic data analysis tools are still executing serially, thus they can not effectively make use of the multi-core architecture to help improve performance, and this also leads to a serious waste of computing resources at the same time; On the other hand, due to the limitations of previous design and development, the interface of the underlying genetic data analysis library used by some tools is unable to take both high performance and user-friendly interfaces into account.

There are some recent studies relating to optimizing and accelerating genetic analysis algorithms, including software optimizations on CPU and GPU platforms as well as hardware accelerations with FPGA. However, most of those studies focused on accelerating a single tool or even a part of a certain tool, while our work targeted to a whole gene data analysis process composed of several tools which can be used in practical clinical detection. "Churchill" made optimizations on the whole process, but it targeted at the GATK-based process while our research object is Gene Panel process whose function, inputs and component tools are all different from those of GATK-based processes.

Based on the Gene Panel process, we designed and implemented a novel Gene Panel data analysis process with massive data-level and task-level parallelism. The new parallel Gene Panel process proposed in this paper achieves a 4～7 times overall speedup over the original process.

This research is supported by the National Program on Key Research Project (2016YFB0200300, 2016YFB0201305, 2016YFB0200504, 2016YFB0200803, 2016YFB0200204) and the Strategic Priority Research Program (XDB24050300).