基于MapReduce的基因读段定位改进算法

涂金金 杨 明 郭丽娜
(南京师范大学计算机科学与技术学院 南京 210023)

摘 要 由于高通量测序技术产生了海量基因读段数据,并行的基因读段定位算法成为近年来的研究热点。对基因匹配算法进行研究,提出了一种基于MapReduce的基因读段定位改进算法,并且通过在读段定位过程中融入生物信息以及利用Hadoop分布式缓存机制,在一定程度上降低了算法的复杂度。在拟南芥菜基因数据集上进行的实验表明,该算法能够有效提高算法执行效率,减少算法执行时间。

关键词 读段定位,MapReduce,SeqMap
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.8.018

Improved Gene Read Mapping Algorithm Based on MapReduce

TU Jin-jin YANG Ming GUO Li-na
(School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China)

Abstract Parallel read mapping algorithms become a hotspot in recent years, since the high-throughput sequence technology generates massive reads. Genetic matching algorithm was studied and an improved gene read mapping algorithm which could reduce the complexity of the algorithm by using Hadoop distributed cache mechanism and integrating biological information was proposed. The experimental results on the Arabidopsis gene data sets show that the proposed improved algorithm can effectively improve the algorithm efficiency and reduce the algorithm running time.

Keywords Read mapping, MapReduce, SeqMap

1 引言

基因读段定位[1-3]是生物信息分析的必要前提,因此基因读段定位算法成为生物信息学研究的热点问题。新一代高通量DNA测序技术[4]的快速发展,产生了海量的基因读段数据,原有串行读段定位算法的执行受到很大的限制,近年来并行基因读段定位算法的研究受到了广泛关注。

目前,并行基因读段定位算法已有了大量的研究成果。在Homer等人提出的BFAST[5]软件中,通过并行多核的程序设计,加速了比对过程。Smith等人利用并行编程实现并改进了Smith-Waterman动态规划[6]的比对方法,加速了读段定位。

MapReduce[7]是一种当前比较流行的并行框架,主要包含map和reduce两个阶段。键值对输入到map函数中,产生一个或多个中间结果,仍以键值对方式输出。reduce函数将拥有相同键的键值对整合在一起,得到最终结果。在MapReduce并行框架下,并行的序列比对研究也取得了一些进展。文献[8]将序列比对算法BLAST和MapReduce结合起来,提出了并行的MapReduce-BLAST算法。Schatz等人也基于MapReduce框架开发了CloudBurst[9]软件,来高速地进行读段数据定位。本文作者前一阶段工作提出的基于MapReduce的考虑剪切位的空位种子索引算法(PJuncSeqMap)[10],将MapReduce和SeqMap中的空位种子索引算法相结合,能够有效地解决海量基因读段数据的定位问题。本文针对其中的基因匹配算法进行改进,并且在读段定位过程中融入生物信息,以及利用Hadoop分布式缓存机制,提出一种基于MapReduce的基因读段定位改进算法。在拟南芥菜基因数据集上进行实验,结果表明,改进算法虽在定位敏感度即正确定位程度上略微降低,但执行时间和效率却大大优于原始算法。

2 PJuncSeqMap算法思想

PJuncSeqMap算法是一种采用SeqMap软件中空位种子索引思想,考虑跨剪切位的情况,并且基于MapReduce框架实现的并行基因读段定位算法[10]。该算法不依赖于由能够定位的读段事先创建出的剪切位点库,不需要借助任何剪切位点信号信息,能获得良好的定位结果。

算法思想描述如下:

//Map阶段
输入:读段数据集,参考序列数据集;
输出:参考序列名称和定位数量。
将每一条读段分割成指定数量不重叠的子读段。
遍历所有读段,对每一个子读段创建种子模式,并将种子与读段关系存入Hash表。

到稿日期:2014-06-05 返修日期:2014-08-24 本文受国家自然科学基金(61272222,61003116),江苏省自然科学基金重点重大专项(BK2011005),江苏省自然科学基金(BK2011782),江苏省普通高校研究生科研创新计划项目(CXLX12_0415)资助。
涂金金(1988-),男,硕士生,主要研究方向为机器学习、模式识别,E-mail:tujinjin1988@sina.com;杨 明(1964-),男,博士,教授,主要研究方向为机器学习、模式识别,E-mail:myang@njnu.edu.cn;郭丽娜(1989-),女,硕士生,主要研究方向为机器学习、模式识别。

遍历参考序列。

1. 搜索读段构建的 Hash 表，首先完成能够不跨越剪切位的定位；

2. 使用跨越剪切位算法对未定位的子读段进行定位；

3. 将分散的子读段组装成完整的读段。

输出参考序列名称和定位数量中间结果。

//Reduce 阶段

输入：所有 Map 输出的中间结果；

输出：参考序列名称和定位数量。

遍历中间结果的参考序列名称，统计各个参考序列的定位数量。

输出参考序列名称和定位数量最终结果。

其中的关键步骤介绍如下：

Map 阶段，步骤 2 使用跨越剪切位算法对未定位的子读段进行定位。根据图 1[10]来描述这一步骤，跨越剪切位读段定位可以分为两种情况：1）未定位子读段的上游和下游子读段都已经定位，如未定位的子读段是 t_2，并且 t_1 和 t_3 已经定位，只需要在 t_2 中寻找一个剪切位点，在满足错配条件下，使 t_2 正好定位于 t_1 和 t_3 之间；2）未定位子读段的上游和下游子读段中只有一个已经定位，如未定位的子读段是 t_4，已定位子读段是其上游子读段 t_3，通过截取 t_4 的前 h 个字符（h-mer），在 t_4 的下游寻找 h-mer 的定位，将其转化为第一种情况进行处理。只有下游子读段已经定位的情况则与上述情况类似。

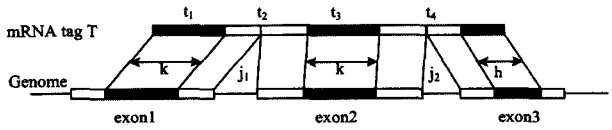

图 1　读段定位示意图

3　改进的 PJuncSeqMap 算法

3.1　Hadoop 分布式缓存的应用

Hadoop 分布式缓存[11]主要是用来将文件分布到集群中所有的节点上，通常是用于分发包含所有 mapper 所需要的“背景”数据的文件。这种机制主要适用于存在一个较大的数据源，而另一个数据源（背景数据，能够存入内存）相对较小，且需要复制到各个节点的情况。

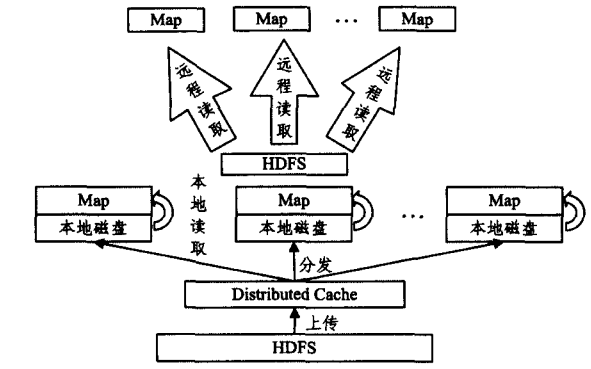

图 2　采用分布式缓存机制与不采用分布式缓存机制读取“背景”数据对比示意图

对于“背景”数据，可以使用分布式缓存文件传递机制，先将这些文件传送到分布式缓存中，然后各个节点从分布式缓存中将这些文件复制到本地的文件系统中使用。具体使用时，各个节点就可以直接从本地文件系统将数据读入内存，减少数据的网络传输时间，从而提高了访问速度。采用分布式缓存机制与不采用分布式缓存机制读取“背景”数据的区别如图 2 所示。本文中，读段数据集相当于较大的数据源，而参考序列数据集相对较小，并且每个 map 在处理过程中都需要使用，相当于“背景”数据，因此本文的这种数据模式特别符合 Hadoop 分布式缓存机制的应用场合。相比于直接从 hdfs 中读取参考序列数据集，采用分布式缓存机制在数据访问效率上有了极大提高。

3.2　融入生物信息缩小搜索空间

第 2 节算法描述中跨越剪切位子读段进行定位是该算法的关键步骤。定位的第二种情况：未定位子读段的上游或下游子读段只有一个已经定位，是普遍存在的。而其中 h-mer 在上游或下游子读段中的搜索空间大小将直接影响算法的效率。在不考虑任何生物信息的情况下，h-mer 需要搜索上游或下游子读段的整个空间。

已知的哺乳动物基因组中只有非常少的内含子长度大于 400kB[12]，因此利用这一生物信息，在定位过程中限定可能的内含子的长度，即上游或下游子读段的搜索空间要小于某个特定的数值。这个数值设定越大，定位结果的敏感性越高，但是定位结果的假阳性率也上升得越快。因此，本文中设定内含子最大长度为 300bp，从而上下游子读段的搜索空间也不会超过 300bp。

因为实验中使用的读段长度小于等于 84bp，在定位过程中，将允许读段跨越至多两个剪切位点。这一生物信息，有助于在第 2 节算法描述中不跨越剪切位的定位完成之后，排除跨越多余 2 个剪切位点定位的情况。这也能在一定程度上缩小读段定位的搜索空间，加快算法执行速度。

3.3　改进的跨剪位匹配算法

本文从跨越剪切位读段定位的两种情况着手，对跨越剪切位匹配算法进行改进。

第一种情况：未定位子读段的上下游子读段都已经定位。假设上游读段定位的末点即未匹配子读段的起始点记为 S，下游子读段定位的起始点即未匹配子读段的末点记为 E。剪切位点记为 M，未定义子读段长度记为 N。原始算法确定剪切位点示意图如图 3 所示。原始算法中具体步骤描述如下：

for M=S+1 to E−1；

将子读段 SM 和 ME 分别匹配至参考序列，记录错配个数；

end；

将错配个数最小的 M 记为剪切位点。

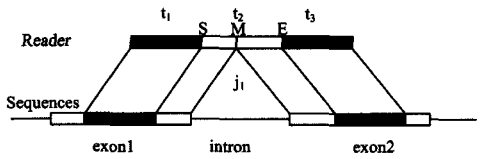

图 3　原始算法确定剪切位点示意图

该算法由于外层循环需要遍历 SE 读段中的 N 个碱基，内层循环需要将 SM 和 ME 分别与参考序列进行比对，因此其算法复杂度为 $O(N^2)$。改进的算法只需遍历 SE 一次即可找出剪切位点，将算法复杂度降为 $O(N)$，具体的算法步骤描述如下（默认允许错配 2 个）：

for I=S+1 to E−1；

从 S 端向 E 端匹配，分别记录错配 0、1、2 的前一个位置，反之从 E 端

向 S 端匹配记录下错配 0、1、2 的前一个位置；

end；

在记录下的位置中查找能够连接的且错配最少的位置，记为剪切位点。

第二种情况：未定位子读段的上下游子段只有一个已经定位。原始算法中，h-mer 固定取值为 2，这使得 h-mer 能够成功匹配到与上游或下游子读段的多个位置，导致未定位子读段的多次尝试匹配，从而降低了匹配效率。h-mer 的大小在很大程度上会影响到这种情况下子读段的定位效率。因此在改进算法中采用动态确定 h-mer 大小的策略。在匹配过程中，主要依赖于已经定位好的上游或下游子读段，如图 4 所示。假设 t_2 为未匹配读段，记长度为 N，t_1 为已匹配读段，预先从 S 点向后匹配，直至记录错配 2 个的位置 M，记 SM 的长度为 L，则 h-mer 的大小可以动态地设为 $N-L$。这样能够尽可能减少未定位子读段尝试匹配次数。在确定 h-mer 的情况下，利用第一种情况下的匹配算法，这样使得改进算法将时间复杂度从原来的 $O(N^3)$ 降低至 $O(N^2)$。

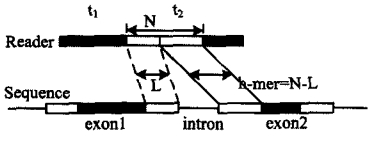

图 4　动态确定 h-mer 大小示意图

3.4　改进的 PJuncSeqMap 算法

利用 Hadoop 分布式缓存机制和生物信息，并且对原算法关键步骤（跨越剪切位的读段定位）进行改进之后，一种改进的 PJuncSeqMap 算法描述如下。

//Map 阶段

输入：读段数据集，参考序列数据集；

输出：参考序列名称和各个节点的定位数量。

1. 利用 Hadoop 分布式缓存机制，从各个 map 节点本地文件系统读取参考序列数据集；
2. 将每一条读段分割成指定数量不重叠的子读段；
3. 遍历所有读段，对每一个子读段创建种子模式，并将种子与读段关系存入 Hash 表；
4. 遍历参考序列；
   - 4.1　搜索读段构建的 Hash 表，首先完成能够不跨越剪切位的定位；
   - 4.2　根据 3.2 节和 3.3 节所述的改进方案对跨越剪切的子读段进行定位；
   - 4.3　将分散的子读段组装成完整的读段；

输出参考序列名称和定位数量中间结果。

//Reduce 阶段

输入：所有 Map 输出的中间结果；

输出：参考序列名称和最终的定位数量。

遍历中间结果的参考序列名称，统计各个参考序列的定位数量；

输出参考序列名称和定位数量最终结果。

4　实验

根据在拟南芥菜基因数据集上进行的实验，对比原始算法，改进算法执行效率更高。

4.1　实验设置与数据集描述

本文实验所使用的云计算平台共有 21 台服务器（1 个主节点，20 个从节点），每个节点配置相同。节点的处理器为 Intel(R) Xeon(R) CPU E5620，主频为 2.40GHz，操作系统为 64 位 Debian Linux 操作系统。Hadoop 平台版本为 hadoop-0.20.2。

从拟南芥菜基因组中抽取 10 条基因作为参考序列，平均长度约为 12000bp。读段数据集主要有两个来源：1）用 ES 培养液培养拟南芥菜，重复实验 3 次，获得 3 次情况下的拟南芥菜 RNA，经测序得到 3 个读段样本集；2）在缺 P 情况下培养拟南芥菜，同样重复实验 3 次，获得 3 个读段样本集。两种情况下获得的读段数据集共 25.2GB，分别称这 6 个读段样本集为 ES1、ES2、ES3、P1、P2 和 P3。

4.2　实验结果分析

第一组实验中读段数据集默认划分为 64MB/块，观察原始算法和改进算法的时间对比。实验结果表明，改进算法能减少算法执行时间，如表 1 所列。

表 1　数据块默认划分情况下，原始算法和改进算法的执行时间(s)对比

算法数据集	ES1	ES2	ES3	P1	P2	P3
PJuncSeqMap	772	721	964	938	958	966
改进的 PJuncSeqMap	668	651	866	854	861	858

第二组实验为了对比原始算法与改进算法在不同规模集群上的执行效率，从 E1 数据集中抽取 1GB 数据，分别在 30、40、50、60 个 map 节点上进行实验，结果表明改进的 PJuncSeqMap 算法执行时间随着集群规模的增大而减少。由于 map 数量的增多，每个 map 处理的数据量减少，因此改进的算法相比于原始算法的提升效果随集群规模的增大而减弱。具体情况如图 5 所示。

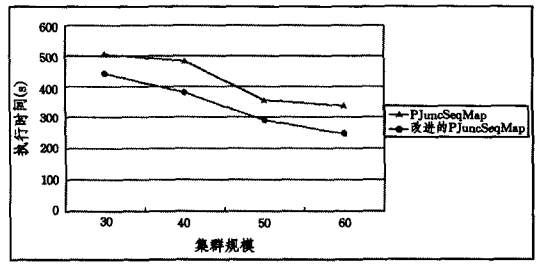

图 5　不同集群规模下原始算法和改进算法执行时间(s)的对比

第三组实验观察在不同大小数据集上原始算法和改进的 PJuncSeqMap 算法的执行效率。实验分别将 300、600、900、1200 万条记录作为读段数据集，参考数据集保持不变，采用 50 个 Map 进行实验，实验结果表明随着读段数据集增大，改进的 PJuncSeqMap 算法与原始算法的执行时间都呈线性增长趋势，具体如图 6 所示。

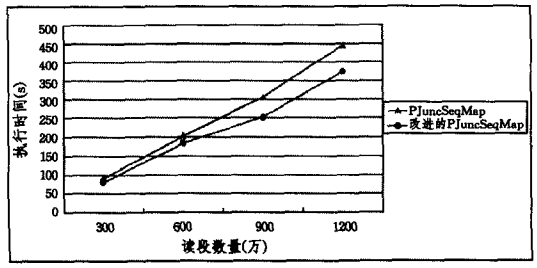

图 6　不同大小数据集上原始算法和改进的 PJuncSeqMap 算法执行效率的对比

第四组实验考察在设置不同大小定位搜索空间，即最大

内含子长度的情况下，原始算法和改进算法在 E1 上获得读段定位数量的对比。实验结果表明，改进算法在有效减少执行时间的同时，并不会大幅度降低定位敏感度，即不会大量丢失正确定位位置。在定位搜索空间设置为 400bp、500bp 的情况下，最终所得的读段定位数量并没有明显变动。因此通过融入生物信息可以选择相对小的搜索空间，从而减少算法执行时间，具体情况如表 2 所列。

表 2　不同大小定位搜索空间情况下，改进算法所得到的读段定位数量对比

参考序列名称 \ 搜索空间大小(bp)	100	200	300	400	500
AT1G03060.1	1612	1634	1640	1647	1648
AT1G05570.1	2794	3037	3043	3045	3049
AT1G27180.1	433	437	437	437	437
AT1G42470.1	394	449	450	450	450
AT1G48090.1	1710	1769	1800	1823	1829
AT1G50140.1	319	346	349	358	369
AT1G56130.1	154	187	189	189	189
AT1G56140.1	713	791	806	806	807
AT1G80070.1	3791	3916	3933	3933	3934
AT1G80410.1	1301	1399	1416	1436	1452

结束语　本文利用 Hadoop 分布式缓存机制，在读段定位过程中融入生物信息，并且对跨越剪切位的匹配算法进行改进，提出一种改进的 PJuncSeqMap 算法。在拟南芥菜基因数据集上进行验证，结果表明改进的 PJuncSeqMap 算法能够有效减少算法执行时间。未来将进一步研究基于 MapReduce 的基因读段定位算法的性能优化问题，寻找到执行效率更高的解决方案。

参 考 文 献

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

[2] Langmead B, Trapnell C, Pop M. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome[J]. Genome Biol, 2009, 10(3): 25

[3] Wang K, Singh D, Zeng Z. MapSplice: accurate mapping of RNA-seq reads for splice junction discovery[J]. Nucleic Acids Res, 2010, 38(18): 178

[4] 王曦，汪小我，王立坤，等. 新一代高通量 RNA 测序数据的处理与分析[J]. 生物化学与生物物理进展，2010，37(8)：834-846
Wang Xi, Wang Xiao-wo, Wang Li-kun, et al. A new generation of high-throughput RNA sequencing data processing and analysis[J]. Progress in Biochemistry and Biophysics, 2010, 37(8): 834-846

[5] Homer N, Merriman B, Nelson S F. BFAST: an alignment tool for large scale genome resequencing[J]. PLoS One, 2009, 4(11): 7767

[6] Smith T F, Waterman M S. Identification of common molecular subsequences[J]. J Mol Biol, 1981, 147(1): 195-197

[7] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters[J]. ACM, 2008, 51(1): 137-150

[8] 杨晓亮. MapReduce 并行计算应用案例及其执行框架性能优化研究[D]. 南京：南京大学，2012
Yang X L. The Application Case Study of MapReduce Parallel Computation and the Optimization of its Runtime Framework [D]. Nanjing: Nanjing University, 2012

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

[10] 涂金金，杨明，郭丽娜. 基于 MapReduce 的基因读段定位算法[J]. 模式识别与人工智能，2014，27(3)：206-212
Tu J J, Yang M, Guo L N. Gene Read Mapping Algorithms Based on MapReduce[J]. Pattern Recognition and Artificial Intelligence, 2014, 27(3): 206-212

[11] 刘鹏. 实战 Hadoop：开启通向云计算的捷径[M]. 北京：电子工业出版社，2011
Liu Peng. Hadoop: open the shortcut to the cloud computing [M]. Beijing: Electronics Industry Press, 2011

[12] 王立坤. RNA-seq 数据的处理与应用[D]. 吉林：吉林大学，2012
Wang Li-kun. Processing and Application of RNA-seq Data[D]. Jilin: Jilin University, 2012

(上接第 51 页)

[9] Xu Yong, Zhang D, Yang Jian, et al. A two-phase test sample sparse representation method for use with face recognition[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2011, 21(9): 1255-1262

[10] Zhang Lei, Yang Meng, Feng Xiang-chu. Sparse representation or collaborative representation: Which helps face recognition? [C]// IEEE International Conference on Computer Vision (ICCV). 2011: 471-478

[11] Mi Jian-Xun. Face image recognition via collaborative representation on selected training samples[J]. Optik-International Journal for Light and Electron Optics, 2013, 124(18): 3310-3313

[12] Feng Zhi-zhao, Yang Meng, Zhang Lei, et al. Joint discriminative dimensionality reduction and dictionary learning for face recognition[J]. Pattern Recognition, 2013, 46(8): 2134-2143

[13] Liu Bao-di, Wang Yu-xiong, Zhang Yu-jin, et al. Learning dictionary on manifolds for image classification[J]. Pattern Recognition, 2013, 46(7): 1879-1890

[14] Li Chun-guang, Guo Jun, Zhang Hong-gang. Local sparse representation based classification[C]// International Conference on Pattern Recognition. 2010: 649-652

[15] Hotta S, Kiyasu S, Miyahara S. Pattern recognition using average patterns of categorical k-nearest neighbors[C]// Proceedings of the 17th International Conference on Pattern Recognition. 2004, 4: 412-415

[16] Cheng Bin, Yang Jian-chao, Yan Shui-cheng, et al. Learning with-graph for image analysis[J]. IEEE Transactions on Image Processing, 2010, 19(4): 858-866

[17] Martinez A M, Benavente R. The AR face database. CVC Technical Report[R]. 1998, 24

[18] Kim S J, Koh K, Lustig M, et al. A method for large-scale-regularized least squares[J]. IEEE Journal on Selected Topics in Signal Processing, 2007, 1(4): 606-617