对其上的差异序列构建索引.当系统中添加了一个新的分布式节点时,系统将对当前负载最高的节点中的分割基准子序列进一步分割,并将分割后的基准子序列和对应其他序列上的差异序列分配给新增加的节点,从而实现高效的扩展.该索引方法可以有效地减少空间开销,但是查询过程中可能会存在匹配序列跨越两个甚至多个节点的情况,直接在每个节点上进行查询可能会有漏解,下面我们将讨论如何在保证高效查询的同时避免查询漏解.

4.2 基于分布式索引的并行查询方法

首先考虑精确查询.由于我们已将基准序列和对应差异序列分配在不同节点上,在查询的过程中可能会出现以下 3 种情况.以图 8 为例,情况 1 是匹配序列完整地出现在一个节点 i 中.这种情况下,我们只需在该分布式节点上独立地进行查询即可.情况 2 是当一个匹配序列跨越了相邻的两个分布式节点时.在该示例中,一个匹配序列跨越了节点 i–1 和节点 i.在该情况下,节点 i 上的查询只能找到部分匹配,剩下的则需要在第 i–1 节点上进一步进行查询.第 3 种情况是对情况 2 的进一步扩展,即当查询串很长时,可能会跨越多个节点.在该示例中,一个查询序列跨越了节点 i、节点 i+1 和节点 i+2,在这种情况下,我们将首先在 i+2 节点上搜索,然后依次在节点 i+1 和节点 i 上进行查询.

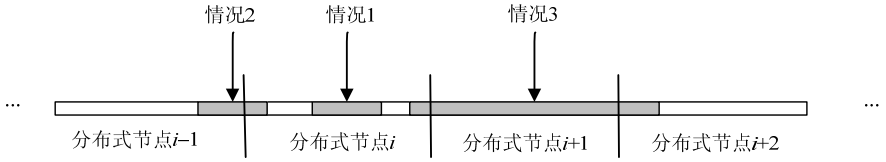


Fig.8 Searching method based on distributed index with segmented reference sequence

图 8 基于分割基准序列的分布式索引的查询方法

下面我们将给出具体的并行查询算法,该方法并行地在每个节点上执行以下操作.对于每个节点 i,它将首先在其上的索引上进行反向搜索,找到一组查询序列 p 的整体或局部匹配序列作为候选集合,见算法 3 第 1 行.如果该过程匹配了完整的查询序列 p,则将其保存在本地的结果集中,否则说明该后缀跨越了当前节点 i,则将其保存在待验证列表中.然后统一发送该列表至节点 i–1,见算法 3 第 7 行.如果该节点为最后一个节点,则发送结束信号给节点 i–1 并返回当前本地结果.否则,就等待节点 i+1 发来的验证列表,并对收到的列表进行验证,并将仍无法验证的部分发送给节点 i–1.当收到节点 i 的结束信号时,该节点将发送结束信号给 i–1 节点,并返回本地结果集,见算法 3 第 16 行~第 17 行.

算法 3. 基于分布式索引的并行查询算法.

输入:分割后的索引 I^S,查询序列 p;

输出: A 为 p 在 S 上的所有精确匹配子串.

在节点 i 上执行以下操作,$1 \leqslant i \leqslant m$

1. 反向搜索 I_i^S 得到候选集 C
2. FOR 候选者
3. IF $c=p$ Then
4. 将 c 添加到本地结果集中
5. ELSE Then
6. 保存 c 到待验证列表
7. 发送待验证列表到节点 i–1
8. IF $i=m$ Then
9. 发送结束信号到节点 i–1
10. 返回本地结果集 A_i
11 ELSE Then

12 WHILE 未收到节点 i+1 结束信号

13. IF 收到验证列表

14. 对其进行验证,并将新的结果保存到结果集 A_i 中

15. 将仍未能验证的候选集发送到节点 i–1

16. 发送结束信号到节点 i–1

17. 返回本地结果集 A_i

该过程可以进一步优化.可以预先通过对节点上的序列进行统计来得到节点上保存的最短序列的长度 $l_{\min}$.如果一个查询序列 p 的长度小于该长度,那么在收到上个节点发送来的验证列表之后就可以提前结束计算过程并返回结果,因为不会有匹配序列可以跨越两个以上的节点.进一步地,如果一个序列 p 的长度满足 $|p|\leqslant k\times l_{\min}$,那么我们可以在收到第 k 次验证列表后提前结束当前计算过程,因为不会有匹配序列可以跨越 k+1 以上的节点.

近似查询与第 3.3 节的过程相类似,首先将序列分割为多个精确查询的序列,并通过上述算法查找其匹配位置,然后在其左右两侧进行扩展验证,与之前算法不同的是,当该序列跨越了不同的节点时,需要首先在右方向查询找到最佳的匹配位置,然后发送该位置和对应最佳编辑距离给左侧节点来进一步验证,对于跨越了多个节点的情况,将重复该过程.最后将为满足编辑距离阈值的匹配返回其最佳匹配位置.采用并行查询方法的时间复杂度为 $O(|p|\times(\log n_{\text{op}}+|C_v|)/m)$.

5 实验与分析

5.1 实验设置

本文实验中的数据来自于两个公开发布的基因序列库:一个是 James Watson 基因库,另一个是炎黄基因库.我们将它们保存为一个基准序列和其他序列与该基准序列的差异.由于任何两个基因序列的差异都在千分之一左右,基准序列的选取对于压缩性能以及查询性能影响不大.本实验采用文献[9]中提到的 UCSC 基因库中的 HG19 作为基准序列.我们采用如下方法构建查询序列.首先在基准序列上随机抽取长度分别为 40、200 和 2 000 的 3 组子序列,每组各 100 个,然后为其添加 1%~5%不等的编辑操作.

本文实验的硬件环境由 20 个节点构成,使用一台 Gigabit 以太网交换机相互连接,每个节点上配置一个 Intel Core 处理器 2.93GHz,4GB 内存,软件环境采用 Linux 操作系统和 C++version4.7.我们主要与文献[9]中提到的基因压缩查询索引方法进行对比,将该方法简称为 GCSI,将本文提出的高效并行的压缩查询索引方法简称为 EPCSI.其中,在 GCSI 方法中,gram 的长度设为 q=10,查询方法采用 C_Verify,该方法为文献[9]中推荐的最佳方法.

5.2 实验结果

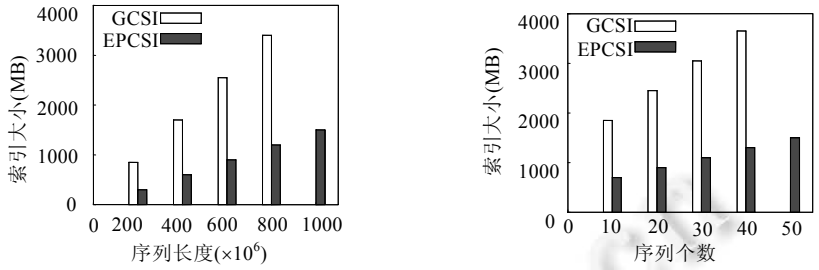
(1) 索引空间开销对比

图 9 对比了两种索引方法的空间开销.其中,图 9(a)对比了在不同序列长度下两种索引的空间开销,我们令查询序列个数都为 50 个,由于采用改进的压缩方法,EPCSI 的索引空间开销远小于 GCSI,特别是在序列长度达到 10^9 时,GCSI 无法在内存中为其构建索引,而 EPCSI 仍可以完成索引构建.图 9(b)对比了在不同序列数目情况下两种方法的索引大小.我们令序列长度为 10^9.同样,在不同序列大小的情况下 EPCSI 的索引空间开销也优于 GCSI,当序列数量达到 50 个时,只有 EPCSI 可以完成索引的构建.在 4GB 内存的单处理机环境下,EPCSI 能够最长处理 2.6GB 的序列.

(2) 精确查询性能对比

图 10 对比了两种方法在查询序列长度分别为 40、200 和 2 000 时的精确查询的时间开销.在查询序列长度为 40 时,GCSI 的性能要优于 EPCSI,但两者的差别不大.但是,随着查询序列长度的增长以及数据集的增加,EPCSI 的性能更好.出现该现象的主要原因如下:首先,随着数据集的增加,EPCSI 的索引空间占用较小,搜索

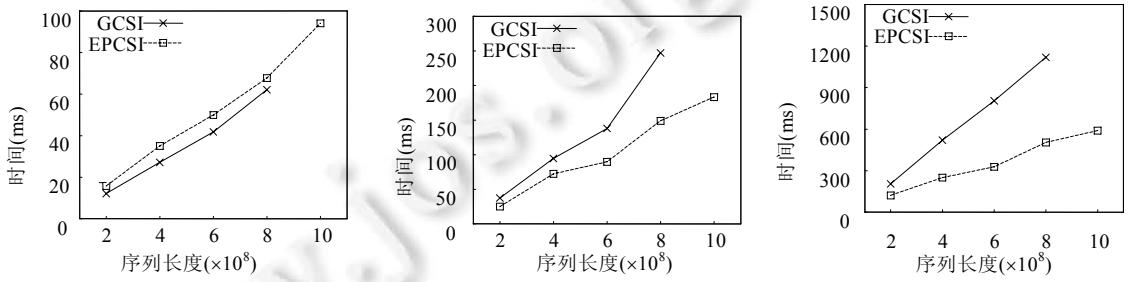
速度更快.其次,GCSI 的 gram 长度预设为 10,随着查询长度的增长,该方法仍需要检查长度为 10 的 gram,无法同时高效地支持短序列和长序列的查询.同样,在序列长度达到 10^9 时,由于无法构建 GCSI 的索引,在该长度下我们只给出进行了 EPCSI 的查询时间.



(a) 不同序列长度下索引大小对比 (b) 不同序列个数下索引大小对比

Fig.9 Comparison of index space cost

图 9 索引空间开销对比



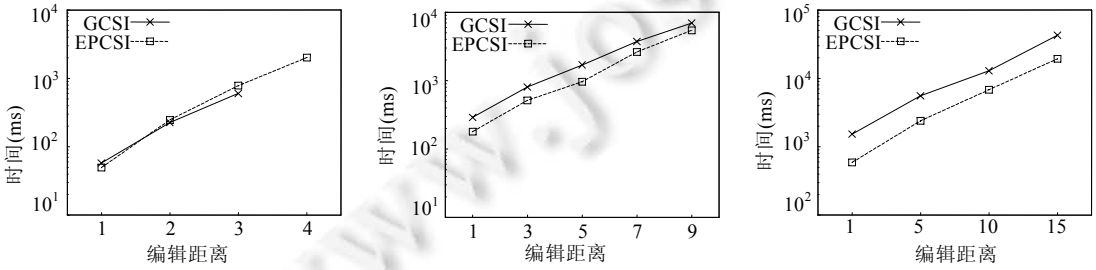
(a) $|p|$=40 (b) $|p|$=200 (c) $|p|$=2000

Fig.10 Performance of exact query search

图 10 精确查询性能

(3) 近似查询性能对比

图 11 对比了两种方法在查询序列长度分别为 40、200 和 2 000 时的近似查询的时间开销.我们令序列的长度为 6×10^8.随着编辑距离的增加,两者查询时间都有所增加,它们的趋势基本一致.在长序列上,EPCSI 的效率更好.该现象的主要原因如下:首先,两种方法都采用了鸽巢原理来对近似序列进行划分.其次,与精确查询相类似,EPCSI 的索引空间更小,查询的效率更高,也更适合长序列查询.并且,EPCSI 在查询过程中利用了共享后缀的方法来提高查询效率.虽然 EPCSI 需要在局部进行解压,在验证时会有额外开销,但在大多数情况下仍优于 GCSI 方法.当查询序列长度为 40、编辑距离为 4 时,每个分割子序列长度为 8,小于预设的 gram 长度 10.在这种情况下,GCSI 无法执行查询.而 EPCSI 没有查询长度的限制,可以正常完成查询.



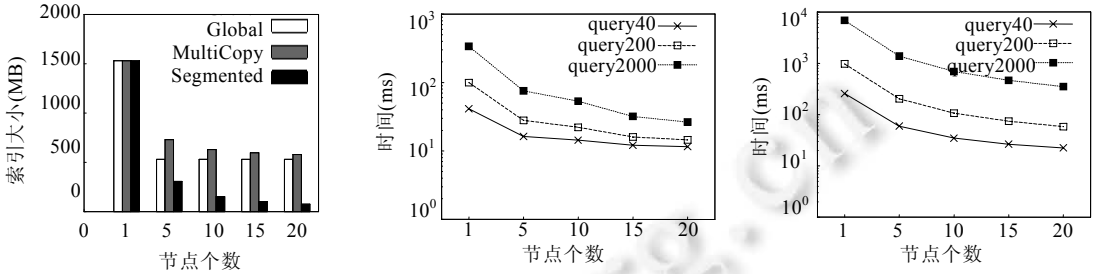
(a) $|p|$=40 (b) $|p|$=200 (c) $|p|$=2000

Fig.11 Performance of approximate query search

图 11 近似查询性能

(4) 并行查询方法性能

图 12 展示了并行查询方法的性能.其中,图 12(a)比较了 3 种不同的并行索引方法的空间开销.随着节点数目的增加,3 种方法的空间开销都有所下降,而其中基于分割基准序列的索引方法的空间开销最小.图 12(b)和图 12(c)分别给出了在基于分割的索引方法上进行精确和近似查询的时间开销.我们分别测试了长度分别为 40、200 和 2 000 的 3 组查询序列.对于近似查询,令各组查询序列的编辑距离分别为 2、5 和 10.可以看到,随着节点数目的增加,查询的时间开销有所降低.



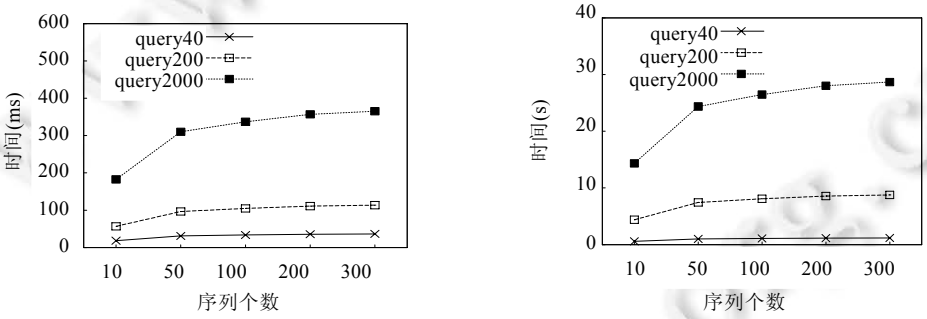
(a) 空间开销 (b) 精确查询 (c) 近似查询

Fig.12 Performance of parallel method

图 12 并行方法的性能

(5) 可伸缩性

图 13 展示了该方法在不同序列个数下的可伸缩性.我们在 20 个节点上对序列长度为 3×10^9 的数据集进行了可伸缩性实验.同样,我们比较了查询长度分别为 40、200 和 2 000 的 3 组查询序列.可以看到,随着序列个数的增加,查询开销并没有线性地增加.该现象进一步说明了基因序列的特殊性,即两个序列之间的差异很小,大多数的查询可以利用共享计算来提高查询效率.



(a) 精确查询 (b) 近似查询

Fig.13 Scalability of the method

图 13 方法的可伸缩性

6 总 结

本文研究了在压缩的生物基因数据上高效并行的查询方法.本文首先对现有的基于基准序列的压缩索引方法进行了改进,对基准序列进行了压缩,进一步减少了空间开销,从而支持更大规模的数据.然后在压缩的数据之上提出了高效的精确查询方法,该方法不需要对查询序列长度有任何限制,可以高效地支持任意长度序列的精确查询.在此基础上,本文对所提出的精确查询方法进一步加以扩展,以支持带有编辑距离的近似查询.然后,本文将现有的串行查询方法进行扩展,提出了空间高效的分布式的索引结构,并且在该索引结构之上提出了并行查询方法.通过在真实的数据集进行实验证实本文提出的方法具有很高的查询效率以及很好的可伸缩性,可以支持更大规模的数据查询.

References:

[1] Mardis ER. The impact of next-generation sequencing technology on genetics. Trends in Genetics, 2008,24(3):133−141. [doi: 10.1016/j.tig.2007.12.007]

[2] Rusk N. Cheap third-generation sequencing. Nature Methods, 2009,6(4):244. [doi: 10.1038/nmeth0409-244a]

[3] Mäkinen V, Navarro G, Sirén J, Välimäki N. Storage and retrieval of highly repetitive sequence collections. Journal of Computational Biology, 2010,17(3):281−308. [doi: 10.1089/cmb.2009.0169]

[4] Wheeler DA, Srinivasan M, Egholm M, Shen Y, Chen L, McGuire A, He W, Chen YJ, Makhijani V, Roth GT, Gomes X. The complete genome of an individual by massively parallel DNA sequencing. Nature, 2008,452(7189):872−876. [doi: 10.1038/nature06884]

[5] Christley S, Lu Y, Li C, Xie X. Human genomes as email attachments. Bioinformatics, 2009,25(2):274−275. [doi: 10.1093/bioinformatics/btn582]

[6] Ferrada H, Gagie T, Hirvola T, Puglisi SJ. Hybrid indexes for repetitive datasets. Philosophical Trans. of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 2014,372(2016):130−137. [doi: 10.1098/rsta.2013.0137]

[7] Kuruppu S, Puglisi SJ, Zobel J. Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval. In: String Processing and Information Retrieval. Berlin, Heidelberg: Springer-Verlag, 2010. 201−206. [doi: 10.1007/978-3-642-16321-0_20]

[8] Kreft S, Navarro G. Self-Indexing based on LZ77. Lecture Notes in Computer Science, 2011,6661(1):41−54. [doi: 10.1007/978-3-642-21458-5_6]

[9] Yang XC, Wang B, Li C, Wang JY. Efficient direct search on compressed genomic data. In: Proc. of the Int'l Conf. on Data Engineering. IEEE, 2013. 961−972. [doi: 10.1109/ICDE.2013.6544889]

[10] Deorowicz S, Grabowski S. Robust relative compression of genomes with random access. Bioinformatics, 2011,27(21):2979−2986. [doi: 10.1093/bioinformatics/btr505]

[11] Wandelt S, Leser U. FRESCO: Referential compression of highly-similar sequences. IEEE/ACM Trans. on Computational Biology & Bioinformatics, 2013,10(5):1275−1288. [doi: 10.1109/TCBB.2013.122]

[12] Brandon MC, Wallace DC, Baldi P. Data structures and compression algorithms for genomic sequence data. Bioinformatics, 2009,25(14):1731−1738. [doi: 10.1093/bioinformatics/btp319]

[13] Daily K, Rigor P, Christley S. Data structures and compression algorithms for high-throughput sequencing technologies. BMC Bioinformatics, 2010,11(19):514−524. [doi: 10.1186/1471-2105-11-514]

[14] Rasko L, Hideaki S, Martin S. The sequence read archive. Nucleic Acids Research, 2011,39(2):19−21. [doi: 10.1093/nar/gkq1019]

[15] Pinho AJ, Diogo P, Garcia SP. GReEn: A tool for efficient compression of genome resequencing data. Nucleic Acids Research, 2011,40(4):27−53. [doi: 10.1093/nar/gkr1124]

[16] Zhu YY, Xiong Y. DNA sequence data mining technique. Ruan Jian Xue Bao/Journal of Software, 2007,18(11):2766−2781 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2766.htm [doi: 10.1360/jos182766]

[17] Claude F, Farina A, Martínez-Prieto MA, Navarro G. Compressed *q*-gram indexing for highly repetitive biological sequences. In: Proc. of the BioInformatics and BioEngineering. IEEE, 2010. 86−91. [doi: 10.1109/BIBE.2010.22]

[18] Schneeberger K, Hagmann J, Ossowski S, Warthmann N, Gesing S, Kohlbacher O, Weigel D. Simultaneous alignment of short reads against multiple genomes. Genome Biology, 2009,10(9):98−119. [doi: 10.1186/gb-2009-10-9-r98]

[19] Claude F, Fariña A, Martínez-Prieto MA, Navarro G. Indexes for highly repetitive document collections. In: Proc. of the 20th ACM Int'l Conf. on Information and Knowledge Management. ACM, 2011. 463−468. [doi: 10.1145/2063576.2063646]

[20] Huang S, Lam TW, Sung WK, Tam SL, Yiu SM. Indexing similar DNA sequences. In: Algorithmic Aspects in Information and Management. Berlin, Heidelberg: Springer-Verlag, 2010. 180−190. [doi: 10.1007/978-3-642-14355-7_19]

[21] Navarro G. A guided tour to approximate string matching. ACM Computing Surveys, 2001,33(1):31−88. [doi: 10.1145/375360.375365]

[22] Lin XM, Wang W. Set and string similarity queries: A survey. Chinese Journal of Computers, 2011,34(10):1853−1862 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01853]

[23] Danek A, Deorowicz S, Grabowski S. Indexing large genome collections on a PC. Eprint Arxiv, 2014,9(10):52−59.

[24] Ferrada H, Gagie T, Hirvola T, Puglisi SJ. AliBI: An alignment-based index for genomic datasets. ArXiv, 2013,7(4):32−39.

[25] Wandelt S, Leser U. MRCSI: Compressing and searching string collections with multiple references. Proc. of the VLDB Endowment, 2015,8(5):461−472. [doi: 10.14778/2735479.2735480]

[26] Burrows M. A block-sorting lossless data compression algorithm. Digital SRC Research Report, 1994,57(4):425−449.

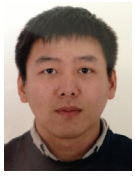
[27] Navarro G, Mäkinen V. Compressed full-text indexes. ACM Computing Surveys, 2007,39(1):2−53. [doi: 10.1145/1216370.1216372]

[28] Lam TW, Li R, Tam A, Wong S, Wu E, Yiu SM. High throughput short read alignment via bi-directional BWT. In: Proc. of the 2009 IEEE Int'l Conf. on Bioinformatics and Biomedicine. IEEE Computer Society, 2009. 31−36. [doi: 10.1109/BIBM.2009.42]

附中文参考文献:

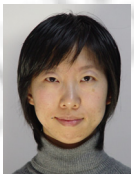
[16] 朱扬勇,熊赟.DNA 序列数据挖掘技术.软件学报,2007,18(11):2766−2781. http://www.jos.org.cn/1000-9825/18/2766.htm [doi: 10.1360/jos182766]

[22] 林学民,王炜.集合和字符串的相似度查询.计算机学报,2011,34(10):1853−1862. [doi: 10.3724/SP.J.1016.2011.01853]



王佳英(1985－),男,山东烟台人,博士生,CCF 学生会员,主要研究领域为字符串精确,近似匹配.

王斌(1972－),男,博士,副教授,主要研究领域为查询优化,分布式数据管理.



杨晓春(1973－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与技术,数据质量分析.