软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(Suppl.(2)):119−126

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

“天河二号”上一种 DNA 序列 de novo 拼接方法的并行优化策略*

张 峰[1], 廖湘科[1], 彭绍亮[1], 朱小谦[1], 王丙强[2], 崔英博[1]

[1](国防科学技术大学 计算机学院,湖南 长沙 410073)

[2](深圳华大基因研究院,广东 深圳 518083)

通讯作者: 彭绍亮, E-mail: pengshaoliang@nudt.edu.cn, http://www.nudt.edu.cn

摘 要: 基于 String Graph 理论的序列拼接工具 SGA 是当前国际上的一种新型序列拼接工具.首先,形式化证明了 SGA 的序列拼接问题是一个 NP 完全问题,然后对 SGA 的拼接效率进行了分析,发现与业界同类拼接软件相比,SGA 在内存开销方面具有优势,但却有更大的时间开销,其中构建索引占了 60%~70%的比例.基于此,设计了一种并行优化策略,并实现了面向天河二号体系结构的并行策略来解决这一问题.分别在普通机群和天河二号上进行性能测试,针对小规模数据,优化后的索引构建时间比之前的最佳性能提高了 3.06 倍,中等规模数据提高了 1.60 倍,实验结果表明,其优化效果明显,且并行构建局部索引过程具有良好的线性扩展性.其中用到的优化方法和策略对相关问题的研究有一定的借鉴意义.这也表明,天河二号的超级计算能力能够很好地助力生命科学领域的相关研究.

关键词: NP 完全问题;并行优化;天河二号;de novo 拼接

中文引用格式: 张峰,廖湘科,彭绍亮,朱小谦,王丙强,崔英博.“天河二号”上一种 DNA 序列 de novo 拼接方法的并行优化策略.软件学报,2014,25(Suppl.(2)):119−126. http://www.jos.org.cn/1000-9825/14030.htm

英文引用格式: Zhang F, Liao XK, Peng SL, Zhu XQ, Wang BQ, Cui YB. Parallel optimization strategy on Tianhe-2 supercomputer for a method of DNA sequence de novo assembly. Ruan Jian Xue Bao/Journal of Software, 2014,25(Suppl.(2)): 119−126 (in Chinese). http://www.jos.org.cn/1000-9825/14030.htm

Parallel Optimization Strategy on Tianhe-2 Supercomputer for a Method of DNA Sequence de novo Assembly

ZHANG Feng[1], LIAO Xiang-Ke[1], PENG Shao-Liang[1], ZHU Xiao-Qian[1], WANG Bing-Qiang[2], CUI Ying-Bo[1]

[1](College of Computer, National University of Defense Technology, Changsha 410073, China)

[2](Shenzhen Huada Gene Research Institute, Shenzhen 518083, China)

Corresponding author: PENG Shao-Liang, E-mail: pengshaoliang@nudt.edu.cn, http://www.nudt.edu.cn

Abstract: SGA is a tool based on string graph theory for DNA sequence de novo assembly. In this paper, the sequence de novo assembly problem based on SGA is proved to be an NP-complete problem, and detailed analysis on SGA is provided. According to the result, SGA outperforms other similar tools in memory consumption, but cost much more on time in which 60%~70% is spent by index construction. To tackle these issues, this paper introduces a deep parallel optimization strategy, and implements a Tianhe-2 architecture oriented parallel framework. Experiments are carried out on different data sizes on ordinary cluster and Tianhe-2. For data of small size, the optimized solution is 3.06 times as fast as before, and for data of medium size, it’s 1.60 times. The results demonstrate the evident overall improvement and the linear scalability for parallel FM-index construction. This study can be beneficial to the optimization research of other relevant issues, and it also affirms the powerful computing ability of Tianhe-2 as a useful tool in life sciences research.

Key words: NP-complete problem; parallel optimization; Tianhe-2; de novo assembly

* 基金项目: 国家自然科学基金(U1435222, 61272056, 61070041); 广州超算应用研发基金(1488064512003, 7411694292900)
收稿时间: 2013-08-05; 定稿时间: 2014-03-13

对 DNA 序列进行分析,首先需要了解 DNA 序列的排列顺序,尤其是对于一些探索性的、参考序列未知的 DNA 序列,目前业界广泛使用的方式是首先由测序仪测序得到生物体基因组的一系列片断,然后利用计算机技术对片断进行 de novo 拼接(无参考序列的拼接),从而还原生物体的完整基因组.目前在 de novo 片段拼接方面存在很多困难与挑战,特别是随着基因组数据规模的增大、复杂程度的提高,往往产生很大的时间和空间开销.

SGA(string graph assembler)是一种新型序列拼接工具,诞生于世界三大基因组研究中心之一的英国 Sanger 研究院.它基于 String Graph[1]理论,创造性地对序列中的重复序列采取了压缩的表示方法,从而从算法原理上节省了内存开销,但运行时间却相对更长,是当前业内研究的热门领域.本文即是从 SGA 出发,分析其性能方面的主要优势和瓶颈,设计并实现了针对性能瓶颈的并行优化方法,从而从整体上提高序列拼接的时空效率,并分别在普通机群和天河二号超级计算机系统上进行了性能测试.

本文第 1 节介绍序列拼接领域的相关工作.第 2 节对 SGA 的性能表现进行定量分析.第 3 节针对 SGA 的性能瓶颈设计并实现并行的优化方法.第 4 节分析"天河"系列超级计算机上的实验结果.第 5 节对全文进行总结.

1 问题描述

1.1 基因序列拼接问题

给定一个 DNA 片段的集合 F, 该集合中的每个元素都是某 DNA 序列 S 的子序列,则 DNA 序列拼接就是利用计算机技术从 F 中重构 S [2].在生物信息学中,通常把随机打断 DNA 序列所得的小片段称作 read.拼接的目标就是利用 read 之间的重叠信息把 read 组装起来,连接成尽可能长的 DNA 序列.这里把拼接最终得到的长序列称为 contig,再由 contig 进一步拼接成 scaffold 完成序列拼接.如图 1 所示.

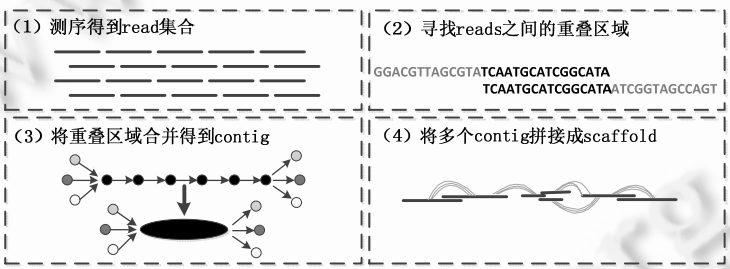

Fig.1 Process of DNA sequence assembly

图 1 DNA 序列拼接过程

1.2 基于SGA的序列拼接问题

1.2.1 问题定义

基于 SGA 的序列拼接主要包括 3 个基本步骤:

(1) 重叠.定义一个阈值,如果两条 reads 之间的重叠区长度超过这个阈值,则认为两条 reads 重叠.

(2) 排列.按照 reads 重叠的先后次序对 reads 排序.

(3) 生成一致序列.找到从起始 read 到终止 read 的一条序列,这条序列包括每个 read 有且仅有一次,则这条序列即为所需的拼接结果.

该序列拼接问题实际上是一个 NP 完全问题,下面给出这一定理的形式化证明.

定理 1. SGA 的序列拼接问题是 NP 完全问题.

证明:首先给出 SGA 的序列拼接步骤:

由于 reads 的数目有限,所以很容易在多项式时间内验证一条路径是否是一条一致性序列,故基于"重叠-排列-生成一致序列"策略的序列拼接问题属于 NP.要证明此问题是 NP 完全的,需要将它归约到一个 NP 完全问题,这里选取哈密顿路径问题并加以描述.

给定一个有向图 $G=(V,E)$,如果能找到一条路径 L,L 经过 G 上的每个节点一次且仅经过一次,则 L 即为 G 上的一条哈密顿路径.我们需要将任意一个哈密顿路径问题的实例转变成上述序列拼接问题的实例,使得哈密

顿问题的回答是肯定的当且仅当序列拼接问题的回答也是肯定的.设 $G=(V,E)$和 L 代表任意一个哈密顿路径问题的实例,我们用顶点代表 reads,用有向边代表重叠关系,如果两条 reads 重叠,不妨假设 v_1 的后缀和 v_2 的前缀重叠,则建立有向边 $e:v_1\rightarrow v_2$,则根据步骤(3),如果能找到从起始 read 到终止 read 的一条序列,这条序列包括每个 read 有且仅有一次,则这条序列即为所需的拼接结果.这样就构成了一个序列拼接问题.所以,如果 G 中存在一条经过每个顶点有且仅有一次的哈密顿路径 L,则序列拼接问题就可以找到一条一致性序列;反之亦然.由于 reads 数目有限,这个归约显然可以在多项式时间内完成.证毕. □

1.2.2 困难与挑战

与业界广泛使用的同类工具相比,SGA 从算法原理上避免了大的内存开销,这是一个非常重要的特点.但是由定理 1 可知,基于 SGA 的序列拼接问题本质上是一个 NP 完全问题,在 SGA 的拼接效率方面,还面临以下问题和挑战:

(1) SGA 的运行时间过长.在对同等规模的数据进行拼接时,SGA 的运行时间往往是其他同类软件的几倍甚至几十倍[3].

(2) SGA 程序的扩展性不好.SGA 本身的实现有一定的并行度,但并没有充分并行化,这一点在第 3 节中会有详细分析;而且本身的并行采用了多线程的实现方式,无法实现跨节点的大规模并行,增加了单节点的计算和内存负载,不能充分利用超级计算机多节点的计算优势.

1.3 相关工作

目前国内外有很多 DNA 序列拼接工具.早期的一些基因拼接软件基于贪心算法设计,这类拼接软件对序列中存在的重复片段不能很好地处理,当一条正在扩展的“种子”遇到由重复序列导致的两条 read 可以扩展时,由于无法区分该如何选择,为了避免拼接错误只能终止扩展,从而导致得到的 contig 很短,影响了拼接效果.基于贪心算法的代表软件有 SSAKE[4],VCAKE[5]和 SHARCGS[6].

当前的许多序列拼接软件使用基于 de Bruijn 图的方法[7].这种方法对测序得到的 DNA 序列处理成有向图,然后在图中找到一条欧拉超路径作为拼接结果[8].主要问题在于处理过程中产生了数倍于原始基因组大小的数据量,由于大量的内存需求无法满足,导致一些大型基因组序列的拼接根本无法完成.基于这种方法的序列拼接软件主要有 SOAPdenovo[9],ABySS[10],Velvet[11]等.

SGA[12]是 2012 年由 Sanger 研究院的 Richard Durbin 和 Jared Simpson 提出的基于 string graph 理论的新型序列拼接工具.这种方法对序列中存在的重复序列(如图 2 中深色部分所示)采取了压缩的表示方式,相对于基于 de Bruijn 图的方法能够显著降低内存开销,而且利用 FM-index 对所有 reads 建立索引,提高了计算重叠区域的效率[13].SGA 方法由于内存开销小,准确度较高,成为了近年来国际上基因拼接领域研究的热点.

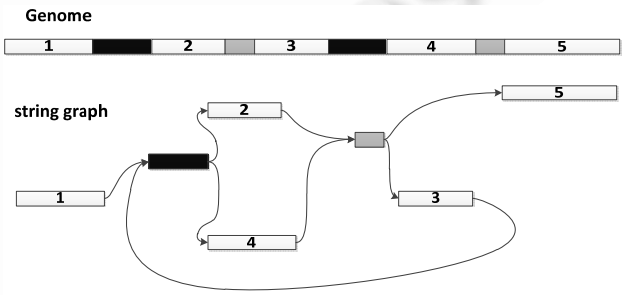

Fig.2 String graph based sequence assembly

图 2 基于 string graph 的序列拼接

2 SGA 性能分析

本文以线虫全基因组的序列拼接为例,对 SGA 的性能从内存开销和时间开销方面进行了动态分析.输入文件为 pair-end 的 Illumina 测序数据,格式为 fastq,大小为 22GB.

2.1 内存开销

如图 3 所示是在对相同基因组序列进行拼接的情况下,SGA 与其他 3 种基于 de Bruijn 图的拼接工具的内存开销比较.由结果可见,在对同等规模数据进行拼接的情况下,其他 3 种工具的内存开销分别为 SGA 的 3.13 倍、5.11 倍、8.62 倍,表明 SGA 在内存开销方面确实更具优势.

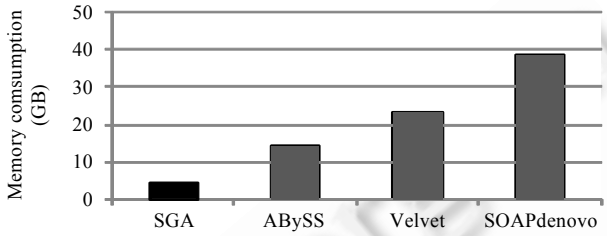

Fig.3 Memory consumption comparison

图 3 内存开销对比

2.2 时间开销

如图 4 所示是在对相同基因组序列进行拼接的情况下,SGA 与其他 3 种基于 de Bruijn 图的拼接工具的时间开销比较.在对同等规模数据进行拼接的情况下,SGA 的运行时间分别为 AbySS,Velvet,SOAPdenovo 的 8 倍、20 倍、3 倍,这表明尽管 SGA 相对来说能够节省内存资源,但它的运行时间却更长.

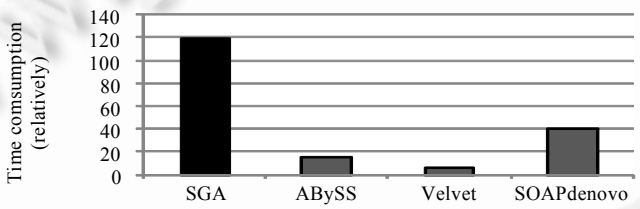

Fig.4 Time consumption comparison

图 4 时间开销对比

我们以大肠杆菌的全基因组序列组装为例,对 SGA 程序进行了动态分析,SGA 的部分过程使用了多线程方法,这里设置线程数为 8,所以部分过程的墙钟时间和 CPU 时间差异较大,见表 1.主要的性能瓶颈步骤在于构建索引(index)过程,索引过程是为了更高效地计算序列之间的重叠而对序列构建 FM-index 的过程,是 SGA 整个工作流程的核心,这一步骤不仅单次执行时间长,而且被多次调用,其执行时间占了整个程序运行时间的 60%~70%.

Table 1 Time breakdown of SGA pipeline

表 1 SGA 流程时间开销分析

步骤	墙钟时间(s)	墙钟时间百分比(%)	CPU 时间(s)	CPU 时间百分比(%)
preprocess	984.68	7.84	984.79	2.53
index	3 132.3	24.95	6 429.46	16.51
correct	1 496.56	11.92	12 801.55	32.87
index	5 687.46	45.31	12 229.89	31.40
filter	577.93	4.60	3 219.36	8.27
overlap	545.47	4.35	3 156.78	8.10
assemble	127.4	1.01	127.6	0.33

2.2.1 热点分析

SGA 本身的实现采用了多线程技术,只能在共享内存多处理器的方式下实现有限的并行,为了避免很大的内存开销,导致只能对每个批次的序列依次串行的构建 FM-index;分批构建 FM-index 后,对中间结果的合并过程也是串行的.

没有充分挖掘索引构建过程中的潜在并行性,从而在很大程度上导致了巨大的时间开销.

2.2.2 可扩展性分析

虽然 SGA 索引构建过程的部分子过程采用了多线程的处理方式,但由于无法实现跨节点的并行处理,无法充分利用像“天河二号”超级计算机这样多节点的计算资源优势,并行的规模有限,可扩展性不好.

3 并行优化方法的设计与实现

根据上述分析,本文设计了多进程结合多线程的混合并行策略:分批构建 FM-index 的过程,任务之间数据耦合度小,依赖关系少,适合多进程的粗粒度并行方式;对分批构建的 FM-index 进行合并的过程,任务之间数据耦合度较大,依赖关系较多,适合多线程的细粒度并行方式.

在这种策略的基础上,本文设计了一个多进程的并行框架,该框架有两个优势:

(1) 并行框架本身可以将分批构建 FM-index 的任务同时分布到多个节点上,利用多个进程同时处理;

(2) 在这个框架的基础上,后续的合并过程可以自然地扩展到多个节点上利用多线程进行细粒度处理,突破了以前不能跨节点的限制,充分利用了计算资源,从而从整体上提高效率.

3.1 并行构建局部FM-index

FM-index 是一种基于 BW 变换[14]和后缀数组的自索引全文查找算法,它将压缩和索引相结合,可以在压缩全文的基础上实现字符串的快速查找而不需要对文件进行解压[15,16],因此可以高效地利用于大文件的压缩和检索[17].FM-index 的构建主要包括 BW 变换和构建后缀数组.

这里,FM-index 的应用有所不同,由前述内容可知,序列拼接需要得到所有 reads 之间的重叠关系,假设 reads 数目为 N,则理论上来说计算两两重叠的时间复杂度为 $O(N^N)$,特别是对于大型基因组的序列拼接,这样的时间复杂度是不可承受的.为了快速地计算 reads 之间的两两重叠,首先对所有的 reads 构建 FM-index,重叠的计算过程就可以利用 FM-index 转化为字符串的查找过程[18],由于 FM-index 可以在 $O(p)$时间复杂度内计算字符串出现的次数,在 $O(p+occ\log\varepsilon u)$时间复杂度内计算字符串出现的位置,因此可以大幅降低计算重叠的时间复杂度.

假设处理的序列数为 N_{seq},进程数为 N_{proc},则进程 i 要处理的序列数:

$$N_i=\begin{cases}N_{seq}/N_{proc}, & \text{if } N_{seq} \bmod N_{proc}=0\\ N_{seq}/N_{proc}+1, & \text{if } N_{seq} \bmod N_{proc}\neq 0 \text{ and } i\leqslant N_{seq} \bmod N_{proc}\\ N_{seq}/N_{proc}, & \text{if } N_{seq} \bmod N_{proc}\neq 0 \text{ and } i> N_{seq} \bmod N_{proc}\end{cases}.$$

这样能保证每个进程处理的序列数最多相差 1,从而最大限度的保证每个进程负载均衡.在这样的任务分配方式基础上,多个进程可以并行构建局部的 FM-index.

3.2 “零通信”的合并预处理

在利用多线程合并上述步骤构建的局部 FM-index 之前,每个线程需要知道自己所对应的文件才能进行合并,称为合并预处理.通常情况下,各个进程需要利用消息通信的方式得到对应的元数据,但这在一定程度上增加了通信开销,特别是随着序列数据规模的增大,构建 FM-index 划分的任务数增多,导致对应的元数据数目也相应增多,通信开销更加不容忽视.

为了减少通信开销,本文重新设计了元数据的构建方式,将元数据和每个进程的 id 建立关系,每个进程维护一个数组 metadata_array,在每轮合并之前,每个进程通过计算得到进程和元数据之间的对应关系,而且保证进程之间 metadata_array 的一致性,这种方法利用并行计算代替消息通信,从而实现了更加高效的“零通信”的合并预处理.

3.3 并行合并局部FM-index

在 SGA 的最初实现中,局部 FM-index 是通过两两合并的方式串行进行的,本文对此进行了改进.在上述并行框架和合并预处理的基础上,每个进程可以知道自己对应处理的文件,因此可以并行地创建多个线程进行两两合并,合并产生的结果作为下一轮的输入,然后更新每个进程的 metadata_array,继续并行的进行两两合并,如此迭代下去,得到最终结果.

4 实验与分析

实验在普通机群和天河二号系统上进行测试,测试平台配置见表 2.

Table 2 Configuration of test platform

表 2 测试平台配置

	普通机群	天河二号
CPU 型号	Intel(R) Xeon(R) CPU E7520 @ 1.87GHz	Intel(R) Xeon(R) CPU E5-2692 v2 @ 2.20Hz
单节点 CPU 规模	16×4	12×2
单节点内存	32GB	64G
节点数	1	16 000
操作系统	Linux version 2.6.18-164.el5 Red Hat 4.1.2-46	Linux version 2.6.32-220.e16 Kylin Linux
硬盘大小	3.6TB	全局共享的并行存储系统,12.4PB
编译器	gcc version 4.1.2 20080704	gcc version 4.4.6 2.11.731
优化选项	-O3	-O3
并行环境	openmpi-1.4.3	MPICH 3.0.4

由于原始的 SGA 实现中只利用了多线程的并行方式,导致扩展性差,因此只能在一个节点内利用共享内存的方式实现有限并行.优化采用了多进程结合多线程的方式,可以实现跨界点的更大规模并行,从而可以充分利用多节点的计算资源,降低时间开销.

4.1 性能对比

实验分别选取了小规模数据集和中等规模数据集进行测试,优化前的 SGA 实现了 3 种构建索引的方法,对优化前和优化后的最短时间开销进行比较,实验均在天河二号系统上进行.实验结果如图 5 所示.

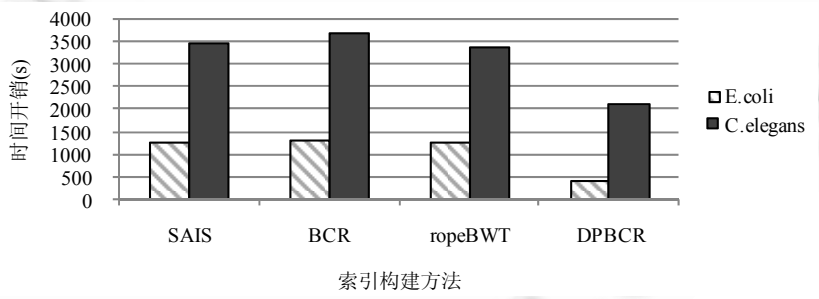

Fig.5 Result comparison

图 5 优化前后结果对比

由图 5 可以看出,优化前利用 ropeBWT 方法构建 FM-index 最快,针对 E.coli 和 C.elegans 样本,分别需要 1 271.79s 和 3 390.10s,优化后的 DPBCR 方法分别需要 415.5s 和 2 122.91s,分别提高了 3.06 倍和 1.60 倍.

4.2 扩展性分析

以 E.coli 的全基因组序列拼接为例为例,并行构建局部 FM-index 的时间开销随进程数变化情况见表 3.

Table 3 Time consumption of parallel FM-index construction

表 3 并行构建局部 FM-index 的时间开销

进程数	1	2	4	8	16	32	64	128	256
时间开销(s)	1 213.29	597.98	293.62	143.10	69.30	31.91	14.26	7.14	3.59

加速比如图 6 所示,结果表明,构建局部 FM-index 的时间加速比随进程数增加基本呈线性增长,表明具有良好的扩展性.

同样,以 E.coli 的全基因组序列拼接为例,并行合并局部 FM-index 的时间开销随线程数变化情况见表 4(以 32 个进程为例).

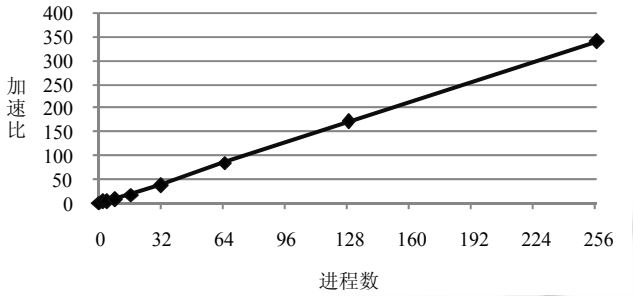

Fig.6 Speedup of parallel FM-index construction

图 6 并行构建 FM-index 过程的加速比

Table 4 Time consumption of parallel FM-index merge

表 4 并行合并局部 FM-index 的时间开销

线程数	1	2	4	8
时间开销(s)	806.31	560.93	453.23	394.56

加速比如图 7 所示,结果表明,随着线程数的增加,并行合并局部索引的时间加速比逐渐变缓,是因为在 SGA 本身实现的合并过程中,每轮合并开始时都要操作上一轮合并产生的临时文件,结束时还要产生临时文件供下一轮合并使用,带来了一定的 I/O 开销,随线程数增加,虽然计算的时间缩短,但 I/O 成为了主要耗时部分,影响了合并过程的扩展性,解决这一问题也是本文的未来工作之一.

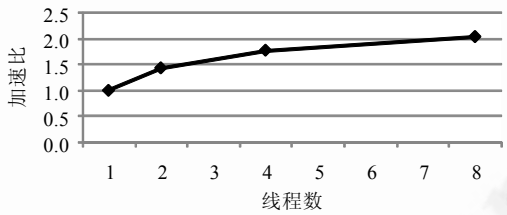

Fig.7 Speedup of parallel FM-index merge

图 7 并行合并 FM-index 过程的加速比

5 总 结

针对新型序列拼接工具 SGA,本文首先形式化证明了 SGA 的序列拼接问题是一个 NP 完全问题,然后对程序进行了分析,由于 SGA 采用了 String Graph 技术对序列中的重复片段压缩表示,降低了内存开销,同时利用 FM-index 提高片段之间重叠区域计算的效率.但总体的拼接时间相对于同类软件却更长,瓶颈在于索引的构建.针对这一问题,本文设计了一种并行优化策略,实现了面向天河二号体系结构的并行策略予以解决.分别在普通机群和天河二号系统上进行性能测试,针对小规模数据,优化后的索引构建比之前的最佳性能提高了 3.06 倍,中等规模数据提高了 1.60 倍.并行构建局部 FM-index 部分达到了线性加速比.实验结果表明,优化效果明显,其中用到的优化方法和策略对相关问题的研究有一定的借鉴意义.另外也证明了天河二号的超级计算能力,能够很好地助力生命科学领域的相关研究.

References:

[1] Myers EW. The fragment assembly string graph. Bioinformatics, 2005,21(Suppl.2):ii79−ii85.

[2] 马文丽,宋艳斌.基因测序实验技术.北京:化学工业出版社,2012.

[3] Salzberg SL, Phillippy AM, Zimin A, *et al*. GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome Research, 2012,22(3):557−567.

[4] Warren RL, Sutton GG, Jones SJ M, *et al*. Assembling millions of short DNA sequences using SSAKE. Bioinformatics, 2007,23(4): 500−501.

[5] Jeck WR, Reinhardt JA, Baltrus DA, *et al*. Extending assembly of short DNA sequences to handle error. Bioinformatics, 2007, 23(21):2942−2944.

[6] Dohm JC, Lottaz C, Borodina T, *et al*. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Research, 2007,17(11):1697−1706.

[7] Earl D, Bradnam K, John JS, *et al*. Assemblathon 1: A competitive assessment of de novo short read assembly methods. Genome Research, 2011,21(12):2224−2241.

[8] Medvedev P, Georgiou K, Myers G, *et al*. Computability of models for sequence assembly. Algorithms in Bioinformatics, 2007: 289−301.

[9] Luo R, Liu B, Xie Y, *et al*. SOAPdenovo2: An empirically improved memory-efficient short-read de novo assembler. GigaScience, 2012,1(1):18.

[10] Simpson JT, Wong K, Jackman SD, *et al*. ABySS: a parallel assembler for short read sequence data. Genome Research, 2009,19(6): 1117−1123.

[11] Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Research, 2008,18(5): 821−829.

[12] Simpson JT, Durbin R. Efficient de novo assembly of large genomes using compressed data structures. Genome Research, 2012, 22(3):549−556.

[13] Simpson JT, Durbin R. Efficient construction of an assembly string graph using the FM-index. Bioinformatics, 2010,26(12): i367−i373.

[14] Mäkinen V, Navarro G. Run-Length FM-index. In: Proc. of the DIMACS Workshop: “The Burrows-Wheeler Transform: Ten Years Later”. 2004. 17−19.

[15] Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. Proc. of the National Academy of Sciences, 2001,98(17):9748−9753.

[16] 李开士,张云泉,李玉成.FM-index 分块并行算法及其实现.计算机工程,2008,34(8):53−55.

[17] 梁军,张迪,张云泉.基于重叠分块的 FM-index 性能研究与分析.计算机工程,2009,35(6).

[18] Hon WK, Lam TW, Sung WK, *et al*. Practical aspects of compressed suffix arrays and FM-index in searching DNA sequences. Institute of Electrical and Electronics Engineers, 2004.

张峰(1989－),男,山东泰安人,硕士,主要研究领域为高性能计算,生物信息学.
E-mail: odysseymtz@gmail.com

朱小谦(1974－),男,博士,研究员,主要研究领域为高性能计算及应用.
E-mail: zhu_xiaoqian@nudt.edu.cn

廖湘科(1963－),男,研究员,博士生导师,主要研究领域为高性能计算,操作系统.
E-mail: xkliao@nudt.edu.cn

王丙强(1977－),男,博士,研究员,主要研究领域为高性能计算,生物信息学.
E-mail: wangbingqiang@genomics.cn

彭绍亮(1979－),男,博士,助理研究员,主要研究领域为高性能计算,生物信息学,移动计算.
E-mail: pengshaoliang@nudt.edu.cn

崔英博(1989－),男,硕士生,主要研究领域为高性能计算,生物信息学.
E-mail: cuiyingbomail@163.com