基于 Pthreads 的并行 DSRC 压缩算法设计与实现

詹 科[1] 张云泉[2] 王 婷[1,3] 郑晶晶[1] 张 鹏[1]
(中国科学院软件研究所并行软件与计算科学实验室 北京 100190)[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]
(广州中国科学院软件应用技术研究所 广州 511458)[3]

摘 要 高通量测序仪产生大量的 DNA 数据,FASTQ 是被广泛使用的存储 DNA 数据的数据格式。对 FASTQ 格式的数据进行压缩处理,能有效地节省存储空间。DSRC 算法具有压缩比高的优点,因此对 DSRC 算法进行并行能提高压缩 FASTQ 格式的 DNA 数据的效率。基于 Pthreads,实现了并行 DSRC 算法。测试结果表明,当使用 4 线程时加速比达到 3.5。

关键词 FASTQ,数据压缩,DSRC,Pthreads
中图法分类号 TP317 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.021

Design and Implementation of Parallel DSRC Compression Algorithm Based on Pthreads

ZHAN Ke[1] ZHANG Yun-quan[2] WANG Ting[1,3] ZHENG Jing-jing[1] ZHANG Peng[1]
(Laboratory of Parallel Software and Computational Science, ISCAS, Beijing 100190, China)[1]
(State Key Laboratory of Computer Architecture, ICT, CAS, Beijing 100190, China)[2]
(Guangzhou ISCAS, Guangzhou 511458, China)[3]

Abstract With the development of high throughput sequencing technology, large volumes of DNA data are being generated. The FASTQ format is widely used to store DNA sequence. If the DNA sequence reads in FASTQ format can be compressed, the storage space will be saved efficiently. One of the DSRC advantages is the high compression ratio, therefore parallel DSRC algorithm will increase the efficiency of compressing the DNA sequence reads in FASTQ format. We implemented the parallel DSRC algorithm based on Pthreads, and the experimental results indicate that the parallel DSRC algorithm gets 3.5 speedup when four threads are used.

Keywords FASTQ, Data compression, DSRC, Pthreads

1 前言

基于下一代测序技术,高通量测序仪产生大量的 DNA 序列数据。存储器的成本虽然不断降低,但是还是不能与序列增长带来的存储开销相匹配。数据压缩可以有效地减少数据冗余,从而节省存储空间。

FASTQ 格式是一种 DNA 序列的常用存储格式。主流的高通量测序平台包括 Illumina GA 和 ABI 公司自主研发的 SOLiD 测序仪。其产生的数据都是以 FASTQ 格式存储的,FASTQ 格式也因此成了高通量测序数据存储事实上的标准格式。并且,FASTQ 格式对 FASTA 格式作如下扩展:添加了与序列中的每个核酸相对应的质量值。以下是 FASTQ 数据的一条记录:

@ERR000625.1 IL16_1138:1:1:1786:371/2
CTTTCCATGGCACTTCTCTGCATACATATTGCTAAC
+
))

每条记录大多数情况下由 4 行数据组成,第 1 行以@起始,是记录的标识符。第 2 行是碱基序,一般是 ATCG 这 4 个碱基重复排列,碱基之间不能有空格。第 3 行一般是单个字符"+",也可以在"+"之后重复第 1 行数据,如 Sanger FASTQ。第 4 行是质量值,与第 2 行的每个碱基字符对应。

存储 FASTQ 格式数据,通常需要占用大量的空间。对于测序深度为 30 层的人基因组原始数据,FASTQ 文件的大小就要 230G 左右。按此推算,5000 个测序样品就会带来 PB 数量级的存储需求。因此,对 FASTQ 格式的数据进行压缩存储,可以有效地节省存储空间。

对于 DNA 序列的 FASTQ 格式,G-SQZ (Genomic SQueeZe)算法[1]由 Tembe W 等实现,是基于 HUFFMAN 算法对 FASTQ 格式的数据进行压缩,高频率字符使用更短的

到稿日期:2013-12-26 返修日期:2014-03-15 本文受国家"八六三"高技术研究发展计划基金(2012AA010903),国家自然科学基金项目(61170075,61133005,61272136,61100073),广州市云计算技术研发与产业化专项资助。

詹 科(1980—),男,博士生,主要研究方向为并行计算,E-mail:zhanke10@iscas.ac.cn;张云泉(1973—),男,研究员,博士生导师,主要研究方向为高性能计算、并行数值软件、并行计算模型、并行数据挖掘,E-mail:yunquan.cas@gmail.com;王 婷(1982—),女,副研究员,主要研究方向为并行算法、高性能计算及性能优化;郑晶晶(1979—),女,助理研究员,主要研究方向为并行计算,E-mail:wangting@iscas.ac.cn;张 鹏(1990—),博士生,主要研究方向为并行计算。

编码，低频率字符使用更长的编码。相对于基准数据集，获得65%到81%的压缩比例。

Quip算法[2]，基于统计模型，使用算术编码，可以压缩FASTQ，SAM/BAM等格式的DNA数据。算术编码的主要优点在于可以使用非整数位对每个字符进行编码。如果某个字符出现的概率是0.1，可以使用理想的编码长度3.3。对于FASTQ的不同行，使用不同的统计模型，这样就能取得更高的压缩比。Quip算法尽管有这些优点，但是实际应用中却由于算术编码的专利权的限制，没有HUFFMAN算法使用广泛。

DSRC算法[3]（DNA Sequence Reads Compressor），用于压缩FASTQ格式数据。DSRC算法比G-SQZ算法有更高的压缩比和更短的压缩时间；比Quip算法有更短的压缩时间。但DSRC算法是串行算法，没有使用多核并行进行数据压缩处理，所以本文对DSRC算法利用Pthreads多核并行进行优化，充分利用多核平台丰富的计算资源，取得更短的压缩时间。

2 利用Pthreads并行DSRC算法

2.1 Pthreads介绍

Pthreads(POSIX threads)[4]，是一套创建和管理线程的应用程序接口。定义了一套C语言的类型、函数和常量的集合。需要pthread.h和thread库支持。在多种类Unix系统（如FreeBSD、NetBSD、OpenBSD，GNU/Linux、Mac OS X和Solaris）上都可以运行Pthreads程序。利用Pthreads，可以充分利用线程的特性，从而编写快捷的软件。

2.2 串行DSRC算法

DSRC分别处理FASTQ格式的3种数据流：标题行、序列行以及质量行。首先将标题行看作是带有分隔符的几个串行字段（分隔符包括空格、冒号等），根据不同的字段特点采用不同的技术（差分编码、直接编码、Huffman编码等）。其次是将序列行内除4个碱基外的其他额外符号从DNA数据转入到质量数据。对于剩余的序列行内的数据，根据DNA碱基数据的统计情况，以及每个碱基数据所对应的质量数据的范围，或者采用直接编码，或者采用Huffman编码。LZ77算法可用于进一步地去除DNA序列中的数据冗余。最后是质量行的压缩，根据质量行数据的特点，首先根据具体情况进行预处理，去除尾部的多个“#”，或者进行游程编码。将预处理之后的数据统一进行Huffman编码。DSRC最大的特点是具有很大的压缩比，在某些应用中压缩比可以超过5:1，并且保持较好的压缩速度。同时DSRC提出一种分层结构，将数据划分为blocks（默认32个records是1个block），512个blocks组织为1个superblock。上述参数可以根据需求灵活设置。每个superblock被独立压缩，1个superblock中的blocks具有共同的统计模型。即，如果要解压缩某个block，首先定位到这个block所属的superblock，计算block在superblock的地址，读取所有blocks共享的统计信息，就可以解压缩block的数据。

2.3 Pthreads优化DSRC算法的方法

DSRC算法的流程如图1所示。

读取数据，当读取的数据量为16384条记录即1个superblock时，调用Process函数进行数据压缩。在执行Process函数的过程中，判断输出到缓冲区的数据大小是否为1MB，如果是，就写压缩数据到文件；如果缓冲区的数据不足1MB，继续执行Process函数的其他指令。当压缩了1个superblock的数据之后，再读取下一个superblock的数据，如此循环，直至所有数据处理完毕。

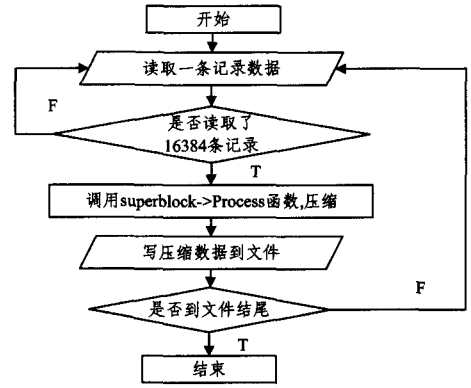

图1 串行DSRC算法流程

并行DSRC算法的流程如图2所示。

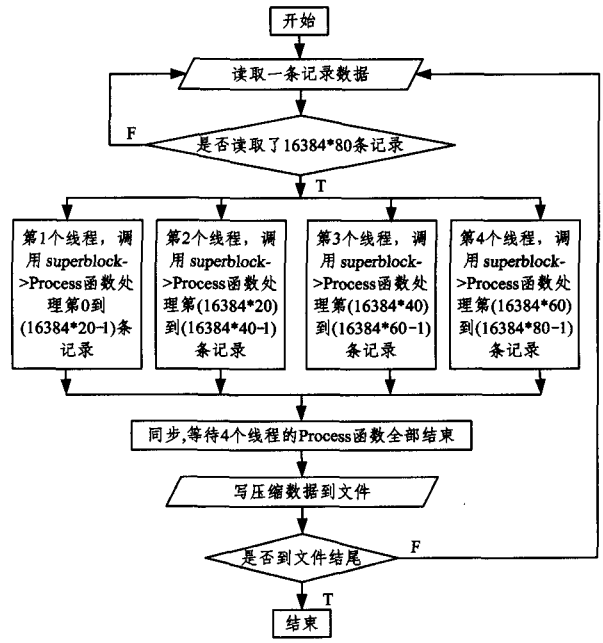

图2 并行DSRC算法流程

DSRC算法的所有压缩指令都在Process函数中执行（Process函数执行时间占全部指令执行时间的71%）。并行算法的思路是开启多个线程，读取一定的数据（16384 * 80条记录，即80个superblocks，测试过其他情形，读取的数据为80个superblocks时加速效果最好，大于80，会因为占用更多的内存导致加速效果更差，小于80，加速效果不明显），多线程同时执行Process函数，以实现压缩指令的并行执行。但是根据串行DSRC算法的流程，当缓冲区的数据有1MB时，写文件，有可能是这样的情况：某个superblock的数据处理结束之后，缓冲区的数据不足1MB，这时候处理下一个superblock，当缓冲区的数据恰好1MB时，写文件。即前后的superblock之间的数据有关联。而并行是以superblock为单位进行处理，需要解决这个问题，才能保证压缩数据的正确。解决的思路是处理完1个superblock，压缩数据存储到缓冲区，

（下转第100页）

International Conference on Software, Telecommunications and Computer Networks (SoftCOM). IEEE, 2011: 1-6
[5] 李凤国. 基于 6LoWPAN 的无线传感器网络研究与实现[D]. 南京: 南京邮电大学, 2013
[6] Gaddour O, Koubaa A. RPL in a nutshell: A survey [J]. Computer Networks, 2012, 56(14): 3163-3178
[7] Gan Wei, Shi Zhi-qiang, Zhang Chen, et al. MERPL: A More Memory-efficient Storing Mode in RPL[C]// International Conference on Networks (ICON). 2013: 1-5
[8] Accettura N, Grieco L A, Boggia G, et al. Performance analysis of the RPL routing protocol[C]// 2011 IEEE International Conference on Mechatronics (ICM). IEEE, 2011: 767-772
[9] Tripathi J, de Oliveira J C, Vasseur J P. A performance evaluation study of RPL: routing protocol for low power and lossy networks[C]// 2010 44th Annual Conference on Information Sciences and Systems (CISS). IEEE, 2010: 1-6
[10] Ko J, Dawson-Haggerty S, Gnawali O, et al. Evaluating the Performance of RPL and 6LoWPAN in TinyOS[C]// Workshop on Extending the Internet to Low Power and Lossy Networks (IP SN2011). 2011
[11] Brachman A. RPL Objective Function Impact on LLNs Topology and Performance[M]// Internet of Things, Smart Spaces, and Next Generation Networking. Springer Berlin Heidelberg, 2013: 340-351
[12] Bloom B H. Space/time trade-offs in hash coding with allowable errors[J]. Communications of the ACM, 1970, 13(7): 422-426
[13] ZigBee Alliance. ZigBee specification. [OL]. 2006. http://www.zigbee.org
[14] Swamidass S J, Baldi P. Mathematical correction for fingerprint similarity measures to improve chemical retrieval[J]. Journal of chemical information and modeling, 2007, 47(3): 952-964
[15] Polastre J, Szewczyk R, Culler D. Telos: enabling ultra-low power wireless research[C]// Fourth International Symposium on Information Processing in Sensor Networks, 2005 (IPSN 2005). IEEE, 2005: 364-369
[16] Levis P, Madden S, Polastre J, et al. TinyOS: An operating system for sensor networks[M]// Ambient intelligence. Springer Berlin Heidelberg, 2005: 115-148

(上接第 91 页)

当每个线程处理完所有的 20 个 superblocks 之后(每次读取的数据是 80 个 superblocks,开启 4 个线程,每个线程处理 20 个 superblocks),写文件。即在 4 线程同步处理 80 个 superblocks 的数据之后再写文件,与串行程序的边压缩边写文件不同。经过验证(将并行程序得到的压缩文件解压缩,与原始的 FASTQ 文件比较,无差异),这种方法能进行正确的压缩。

3 结果和分析

在如下 2 种测试环境中,比较了串行程序与多线程程序压缩不同大小的 FASTQ 文件的 Process 函数运行时间。测试环境 1 为操作系统:Ubuntu 11.04,内核 2.6.38,GCC 版本号 4.5.2,8 核处理器,CPU 型号:Intel(R) Xeon(R),X5550,2.67GHz,内存:16GB。测试环境 2 为操作系统:Ubuntu 10.10,内核 2.6.35,GCC 版本号 4.4.5,8 路 8 核,CPU 型号:Intel(R) Xeon(R) X7550,2.00GHz,内存:512GB。

测试结果如表 1 所列。

表 1 串行、并行 DSRC 压缩时间比较

数据文件[5]	测试环境 1			测试环境 2		
	DSRC (单线程)	Pthdsrc (4 线程)	加速比	DSRC (单线程)	Pthdsrc (4 线程)	加速比
SRR013951_2. filt. fastq (3GB)	78	20	3.9	107	30	3.6
SRR765989_1. filt. fastq (7.3GB)	268	76	3.5	384	110	3.5
SRR741385_2. filt. fastq (21GB)	—	—	—	1155	321	3.6

DSRC:串行 DSRC 算法。Pthdsrc:基于 Pthreads 的并行 DSRC 算法。分别使用测试环境 1 和环境 2 的单线程和 4 线程,得到表 1 所列的结果。时间单位是秒。因为测试环境 1 的内存是 16GB,而 SRR741385_2. filt. fastq 文件是 21GB,所以没有在测试环境 1 下压缩 SRR741385_2. filt. fastq 文件。

在测试环境 1 下,压缩 SRR013951_2. filt. fastq,串行 DSRC 程序执行 Process 函数的时间是 78 秒。使用 2 个处理器核,Process 函数的执行时间是 38 秒,Process 函数的加速比是 2。使用 4 个处理器核,Process 函数的执行时间是 20 秒,Process 函数的加速比是 3.9。对于单个 superblock 的处理时间,在串行 DSRC 中是 0.08 秒,在 Pthdsrc 中是 0.02 秒。并行程序有几乎线性的加速比。

在测试环境 2 下,使用 4 线程分别压缩 3 种大小不同的 FASTQ 文件,相对于串行程序,加速比是 3.5。压缩 SRR765989_1. filt. fastq(7.3GB)文件,使用 8 个处理器核,并行算法的 Process 函数的时间是 185 秒。使用 10 个处理器核,并行算法的 Process 函数的时间是 181 秒,相比使用 4 个处理器核的情况,性能下降。

基于 Pthreads 的并行 DSRC 压缩程序,在测试环境 1 下,使用单线程、2 线程、4 线程,Process 函数有几乎线性的加速比。在测试环境 2 下,使用单线程、4 线程,Process 函数有接近3.5的加速比。当线程数增加到 8 或者 10 时,相对于 4 线程,性能下降。可能的原因是内存的限制以及通信开销的增加抵消了多线程处理带来的速度提升。

结束语 本文以串行 DSRC 算法为研究对象,基于 Pthreads 技术优化 DSRC 算法,测试结果表明,并行 DSRC 算法性能提升。下一步研究将此 Process 函数模块嵌入至进程与线程的混合并行 DSRC 算法优化中,以促进压缩算法并行效率的整体提升。

参 考 文 献

[1] Tembe W, et al. G-SQZ: compact encoding of genomic sequence and quality data[J]. Bioinformatics, 2010, 26(17): 2192-2194
[2] Jones D C, et al. Compression of next-generation sequencing reads aided by highly efficient de novo assembly[J]. Nucleic Acids Res., 2012, 40(22), e171
[3] Deorowicz S, et al. Compression of DNA sequence reads in FASTQ format[J]. Bioinformatics, 2011, 27(6): 860-862
[4] https://computing.llnl.gov/tutorials/pthreads/#Abstract
[5] http://www.1000genomes.org/