超长 DNA 序列的高效压缩算法研究

欧阳继超,冯 萍,康继昌
(西北工业大学 计算机学院,陕西 西安 710072)

摘 要:DNA 序列虽然只由四个碱基组成,但数据量却非常巨大。有效的压缩 DNA 数据能大量节省传输的时间开销。目前已经有一些 DNA 序列专用的压缩算法,如 Biocompress,DNACompress 和 CTW+LZ。虽然这些算法可以获得较好的压缩比,但是由于采用了传统的 CTW 算法或 LZ 系列的字典替换,导致花费太多的时间。为了解决这一问题,提出使用改进的 RLE,差分编码和可变长整形等一系列编码方式进行多重压缩的高效压缩算法 Dzip。标准 DNA Benchmark 数据测试的实验数据表明,该算法与现行 DNA 专用压缩算法相比,加速比至少为 28。

关键词:双序列比对;DNA 数据压缩;可编程门阵列;差分编码;可变长整形
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2013)12-0001-04
doi:10.3969/j.issn.1673-629X.2013.12.001

An Efficient Compression Algorithm for Huge DNA Sequence

OUYANG Ji-chao,FENG Ping,KANG Ji-chang
(School of Computer Science,Northwestern Polytechnical University,Xi'an 710072,China)

Abstract:The DNA sequence is composed of only four base,but has lots of data. The effective compression for DNA data can save much time. There are several DNA sequence oriented compression methods like Biocompress,DNACompress and CTW+LZ. These algorithms can achieve good compression ratio,but has sacrificed too much time searching for similar areas. In order to solve the problem,a new algorithm Dzip was presented,by means of multiple layers compression techniques like improved RLE,delta encoding,variable integers. In comparison with current DNA sequence oriented compression methods,the standard DNA benchmark results indicate that the new algorithm can achieve at least 28 times faster in running time.

Key words:pair wise sequence alignment;DNA data compression;field programmable gate arrays (FPGA);delta encoding;variable integers

1 概 述

DNA 序列虽然只需 4 个碱基{A,T,C,G}即可表示,但数据量却非常巨大。一个完整人类基因组序列约有 32 亿碱基对[1-2]。若使用普通字符编码方式,如此大数据量的传输将十分耗费时间。

该实验室提出并实现基于布尔逻辑的双序列比对协处理器,利用专用的并行加速算法显著提高了双序列比对速度[3]。由于主机与协处理器间传输速度上的瓶颈,使得比对完整人类基因组序列在传输上开销十分巨大。高效的数据压缩可以增大数据传输的吞吐量。因此实现一个高效且高速的压缩和解压算法可有效解决大数据的传输问题。

通用文本压缩方案如 gzip 和 bzip2,可以高速并且高效地压缩文本文件,但却不能高效地压缩 DNA 序列。因为 DNA 序列中只有 4 个碱基 A,G,C,T,用最简单的二进制编码方式只需 2 比特即可对一个碱基进行编码。用通用文本压缩方案,不但不能压缩序列,反而将每个碱基扩展到大于 2 比特。

Scott Christley 推出一种压缩方法,可以使基因组序列大小减少到只有 4 MB[4],小到足以作为电子邮件的附件发送。该方法在针对存储大量基因组序列中会很有效,而用来压缩仅仅一对序列显然是不够高效的。这类技术却需要一个完整的 3 GB 大小的参考基因组[5],和一个 1.2 GB 的 SNP Map。最近也出现了一些基于类似思想来压缩数据库的压缩算法[6]。

目前一些压缩效果较好的 DNA 序列的专用压缩

收稿日期:2013-02-26 **修回日期**:2013-05-28 **网络出版时间**:2013-09-29
基金项目:国家"863"高技术发展计划项目(2003AA001018);航空科学基金(02F53031)
作者简介:欧阳继超(1986-),男,硕士研究生,研究方向为嵌入式计算;冯 萍,教授,研究方向为基于基因的次序;康继昌,教授,博士生导师,研究方向为基于基因的次序。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20130929.1522.012.html

方法,如 Biocompress[7],DNACompress[8],DNABIT[9],基于上下文加权树的 CTW+LZ[10],和使用动态规划的 DNADP[11]。这些压缩方法可以达到较低的压缩比,但是都需要花费太多的压缩时间。这类算法在很长一段 DNA 序列中重复搜索近似区域,虽然可以获得低压缩比,却会使得算法复杂度过高。例如,CTW+LZ 算法使用了复杂的上下文树和 Lempel-Ziv[12]算法,需要花费几个小时来压缩一个仅 227 kB 的 HEMCM-VCG 序列。像这样的牺牲太多时间赢来低压缩比的算法,也是不可取的。

通过分析以上算法的不足,结合对生物数据特点的利用,文中设计和实现了高速 DNA 序列数据压缩算法 Dzip。Dzip 使用一系列简单的编码方法对数据进行多次压缩,在降低压缩算法复杂度的同时,也降低了 FPGA 协处理器端的解压算法复杂度。

2　Dzip 的算法实现

文中使用一套基本的压缩方法 Dzip,使用多种压缩编码技术对序列进行多次压缩,来实现高效的压缩。这些编码技术主要包括改进的 RLE 编码、Delta 编码、VINT 可变长整形编码。

例如现有待比对序列 seqX,seqY,其长度 |seqX| = |seqY| = N。Dzip 压缩算法主要分为两部分完成(如图 1)。

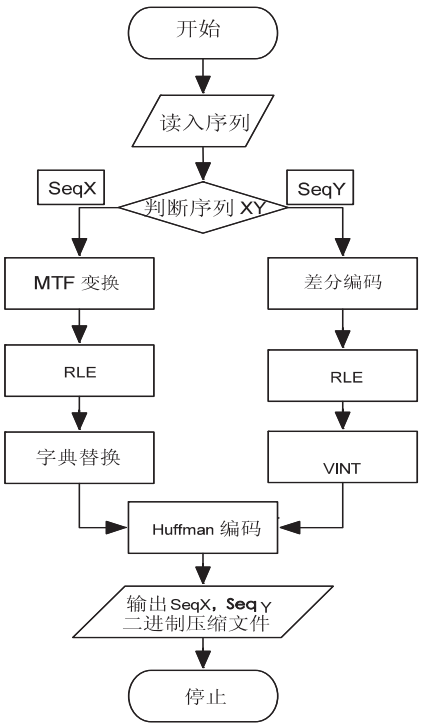

图 1　Dzip 压缩流程图

压缩 seqX 步骤:

1)用 MTF 变换将序列 seqX 转换为更有可能出现连续重复的字符串。

2)使用 RLE 编码消除 MTF 所产生的连续重复。

3)构建高频率重复出现子串的字典,用子串在字典中的排序替换序列中的子串。

4)最后使用 Huffman 编码将字符串转换为二进制编码。

压缩 seqY 步骤:

1)以 seqX 作为参考序列,使用差分编码将 seqY 进行编码。

2)使用 RLE 编码消除差分编码所产生的连续重复字符。

3)使用 Variable Integers(VINT)来记录变换位置。

4)使用 Huffman 编码将字符转换为二进制编码。

此节主要介绍压缩步骤中的各个关键编码方式的实现。

2.1　MTF 变换

前移变换(Move-To-Front,MTF)的主要思想是将字符编码为在“最近使用字符”列表中的索引,索引为该字符在列表中对应的位置。这个“最近使用字符”列表在每次编码后会更新,被编码用过的符号前移至列表首位。

例如有 seqX 为“TATCATGT”,初始列表为“ATCG”,列表的索引为“0123”。其中,A 对应 0,T 对应 1,C 对应 2,D 对应 3。MTF 对该序列的处理过程如表 1 所示。

表 1　MTF 变换

SeqX	MTF 编码	列表
TATCATGT	1	ATCG
TATCATGT	1	TACG
TATCATGT	1	ATCG
TATCATGT	2	TACG
TATCATGT	2	CTAG
TATCATGT	2	ACTG
TATCATGT	3	TACG
TATCATTT	1	GTAC

编码第一个字符“T”,使用“T”在列表“ATCG”中的索引对其进行 MTF 编码,得 1。编码后将“T”前移,更新列表为“TACG”。重复这个步骤直到完成整个序列编码,输出序列将是“11122231”。

显然序列会在 MTF 变换后出现更多的连续重复。而且 MTF 变换也是可逆变换。因此方便对数据进行解压操作,逆变换只需将列表中首字符输出,并且将其插入到编码所指的列表中的位置即可。

但是此项步骤并没有压缩数据,之所以首先使用 MTF 变换的原因,是在变换后的序列更有可能出现长的连续重复区域,从而可以为接下来的 RLE 编码降低压缩比。而且变换步骤简单可逆,也降低了解压算法

的复杂度。

2.2　改进的 RLE

用 RLE(Run-Length Encoding)消除 MTF 的输出序列中连续重复出现的字符。RLE 编码的原始方法是将连续重复的符号序列转换为第一个符号和该重复的长度。例如将“AAAAATTT”编码为“A4T2”。对于原始 RLE 编码产生的字符串,由于多了数字的编码,字符集需要加入 0 至 9 十个数字字符。这样,字符集大小扩展至 14,会导致对字符集的二进制编码提高至 4 比特每字符。

文中提出采用反转的二进制编码,不需要添加 10 个字符,用 P 和 Q 两个特殊的符号来代表 0 和 1。例如 53 的二进制表示是 110101,由于解码操作是从左往右,为了方便解码,将二进制编码反转为 101011,并使用 P 与 Q 替换,得到 QPQPQQ。

因此 MTF 变换产生的序列“11122231”可以由改进的 RLE 编码为“1PQ2PQ31”,这样的数字编码方式比传统的 RLE 更为灵活,可以编码任意长度的整数。假设 MTF 变换所产生的序列中出现连续重复区域个数为 M,平均每个区域有 L 次重复,重复次数的特殊编码长度平均为 K。至此序列的字符集中总共包括 6 个字符{0,1,2,3,P,Q},如果简单地用 3 比特来二进制化每个字符,此时压缩比(bits/base)为:

$$\frac{3(N-M(L-K))}{N}$$

其中,N 为序列长度也就是碱基个数。由公式可以看出,序列中的重复区域 M 越多,特殊编码的长度 K 越长,所产生的压缩比越高。

2.3　字典替换

字典替换法是以类似查字典的方式进行编码。它的基本原理是以经常出现的较长字符串组合构成字典中的各个词条,并用相对较短的数字或符号来替换的方法。经过 RLE 编码消除了由 MTF 变换序列中的所有连续重复区域,产生了一个由 6 位字符集组成的字符串。由于所有的连续重复已经被去除,接下来的所有字符都是 6 个字符集中字符的无连续重复的排列组合。文中采用建立高频率的重复子串字典并进行替换来降低压缩比,步骤如下:

1)用 4 位窗口滑块从左向右滑动扫描序列,记录子串的出现频率。

2)寻找到列表中频率最高的前 4 个子串。

3)用序列号编码替换序列中对应的子串,并在每个序号前加入前缀 V,即用 V 与序列号替换序列子串。

4)同样使用 3 位长的窗口对序列进行扫描。

5)寻找到列表中频率最高的前 4 个子串。

6)使用 U 与序列号替换序列子串。

假设原始 seqX 序列的一个片段为“02132021 0301232012033333”,总长度为 24,经过一次改进 RLE 编码以后,变为“0213202103012320 1203PPQ”,如图 2 所示。假设步骤 2 两次生成的子串频率前 4 位如表 2 所示,那么字典替换执行完后的序列为“$V_2U_0V_0V_320V_0$PPQ”,长度已缩短至 15。设 3 位子串的个数为 C,4 位子串的个数为 D。压缩比可表示为:

$$\frac{3(N-(2D+C+M(L-K)))}{N}$$

表 2　子串字典

出现频率次序	4 位长子串	编码	3 位长子串	编码
0	1203	V_0	202	U_0
1	3210	V_1	102	U_1
2	0213	V_2	213	U_2
3	0123	V_3	123	U_3

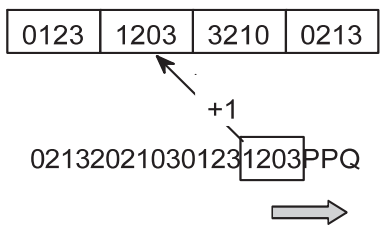

图 2　扫描序列示意图

通过计算上述示例的原始数据压缩后的压缩比为 1.56(bits/base)。

2.4　Huffman 编码

根据字符出现频率构建 Huffman 树,对序列进行二进制编码,用更短的编码代替频率高的字符,降低整个序列的编码长度。对于 RLE 编码的长度为 13 的输出序列“$V_2U_0V_0V_320V_0$QPQ”,总共有 8 个不同的字符组成的字符集 $S=\{0,1,2,3,P,Q,U,V\}$,出现频率 $C=\{0.37,0,0.125,0.125,0.125,0.125,0.125,0.5\}$。那么 Huffman 编码为 $H(S)=\{10,0010,0011,010,011,000,11\}$,每个符号的二进制编码平均长度为:

$$L(H)=\sum_{i=1}^{n}c_i\text{length}(s_i)=2.58$$

其中,c_i 为第 i 个字符的出现频率;s_i 表示第 i 个字符。由于字符集非常小,所生成的 Huffman 树也很小,因此解压过程也比较简单。于是 seqX 的压缩比可以总结为:

$$\frac{\sum_{i=1}^{n}c_i\text{length}(s_i)(N-(2D+C+M(L-K)))}{N}$$

2.5　差分编码

两个人类基因组超过 99% 是相同的,所以压缩 seqY 序列只需存储相对于 seqX 的差分区域(Delta 区域)就可以大大降低压缩比。

差分编码可分为两步进行。首先只存储数据之间

的差异(Delta)区域而不是完整地存储整段序列,这部分采用与压缩 seqX 序列相同的策略。第二步是存储 Delta 的位置,同样可采取 Delta 编码将大数转为较小数字以便于压缩存储。例如图 3 中步骤 a 中的位置{30,33,34,35,36},将只存储为{30,3,1,1,1},然后使用 RLE 编码压缩重复字符(图 3 步骤 b)。由特殊字符 U 和 V 反转二进制编码重复次数。

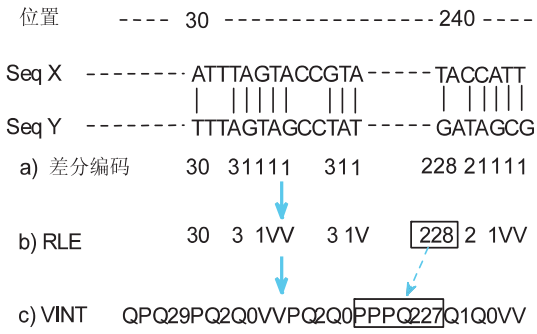

图 3　序列 Y 的 Delta 位置压缩步骤

2.6　可变长整形(VINT)

在改进的 RLE 中对于数字使用反转二进制的方法来进行编码。然而在差分编码中,由于染色体的大小可以有 247 MB,最大位置需要使用 28 比特来进行二进制编码。若使用特殊符号 P 和 Q 的反转二进制编码,会扩大序列二进制长度。

在这里,文中提出可变整数的方法。使用特殊字符 P 和 Q 对数字的位数进行反转二进制编码。例如,对于图 3 中的位置 228,它可以被编码为 PPPQ11100011,其中 PPPQ 将计算得到等于 8,这意味着随后的 8 比特为一个整数。11100011 为 227 的 2 进制编码,在这里将 228 减去 1,使位置由 0 开始编码(图 3 步骤 c),图中为了方便理解使用 10 进制数字进行表示。最后,使用哈夫曼编码对四个特殊字符进行编码。

3　性能分析与对比

实验对比选用压缩比与压缩时间作为评价准则,压缩比是衡量算法压缩效率的指标之一,压缩比越小,说明算法压缩效果越好。同样压缩比的情况下压缩时间越短,说明压缩效率越高。

通过与现行 DNA 序列数据专用压缩算法中效率较好的 GenCompress,DNACompress 和 CTW+LZ 算法进行对比(见表 3 和表 4),表 4 的测试结果显示 Dzip 在压缩时间上大大优于其他算法,对于正常标准测试 DNA 序列的压缩时间都能控制在毫秒级。现有算法如 CTW+LZ 算法压缩仅 227 kB 的 HEMCMVCG 序列需要花费几个小时,但压缩比并没有得到显著改善。Dzip 在压缩比与其他算法不相上下的基础上,将时间降至毫秒级别,与其中最快的 DNACompress 相比加速比至少为 28。

针对 DNA 序列的特点,这种压缩方式可以生成 seqX 序列的压缩文件,以及 seqY 与 seqX 序列的 Delta 文件。当 seqX 与 seqY 相似度高时,由于采用 Delta 编码,seqY 序列的压缩效果尤为明显。由于各编码算法复杂度低,而且编码过程都简单可逆,因此也降低了 FPGA 协处理器端解压缩算法的复杂度,节省 FPGA 的计算资源。整个压缩的所有编码过程都没有改变序列的顺序,这也是文中只使用 MTF 变换来制造更多的重复,而放弃加入 Burrows-Wheeler 变换的原因。因此 FPGA 协处理器端解压时可以将相对很小的 Delta 文件存储于缓存中。当输入已压缩的 seqX 时不必一次解压缩整个文件,可多流水线式传输,使得解压和比对并行工作。

表 3　与现行 DNA 压缩算法压缩比的对比(bits/base)

序列	长度	DNACompress	GenCompress	CTW+LZ	Dzip
HUMDYSTROP	38 770	1.911	1.923	1.918	1.921
HUMHBB	73 308	1.789	1.820	1.808	1.836
MPOMTC	186 609	1.892	1.901	1.900	1.882
HEMCMVCG	229 354	1.849	1.847	1.841	1.842
VACCG	191 737	1.758	1.761	1.761	1.872

表 4　压缩时间对比(Intel Celeron 1.86 GHz CPU)

序列	长度	GenCompress/s	CTW+LZ/min	DNACompress/s	Dzip/ms	最小加速比
HUMDYSTROP	38 770	3	3	2	69	28
HUMHBB	73 308	32	40	3	93	32
HEMCMVCG	229 354	971	150	13	332	39

4　结束语

针对目前一些 DNA 序列算法的不足,文中设计和实现了高速 DNA 序列数据压缩算法 Dzip。Dzip 使用一系列简单的编码方法对数据进行多次压缩,在降低压缩算法复杂度的同时,也降低了 FPGA 协处理器端的解压算法复杂度。

参考文献:

[1] Wheeler D A, Srinivasan M, Egholm M, et al. The complete genome of an individual by massively parallel DNA sequencing[J]. Nature, 2008, 452(7189): 872-876.

[2] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome[J]. Nature, 2004, 431(7011): 931-945.

[3] 王进科,冯　萍,康继昌,等. 基于布尔逻辑的双序列比对协处理器的设计与实现[J]. 西北工业大学学报, 2011, 29(1): 1-5.

[4] Christley S, Lu Y, Li C, et al. Human genomes as email attachments[J]. Bioinformatics, 2009, 25(2): 274-275.

(下转第 10 页)

```
drawpositionY : 450,
showsizewidth : 50,
showsizeheight : 46.5,
frame : [0,0,0,0,0, 1,1,1,1,1,2,2,2,2,2, 3,3,3,3,3],
circle : -1,
direction : "endwise",
filter : ["boxblurfilter",true , [0 ,0 ,0 ]],
type : "Sprite"
};
img1. image. src="back. jpg";
```

上面的代码创建了一个从 1 开始,持续到 4 000 的动画,模仿上面的代码,还可以创建别的动画。按照上述模式,创建了一个有关物流运输的动画,虽然实际编程时,不会真地创建一个时间轴,但是就像文章的中心线一样,时间轴贯穿整个动画,它将各个分动画串联成一个完整的、有意义的动画,为了更直观地展示物流动画中各分动画的关系,绘制了图 4(实际编程不需要创建时间轴)。

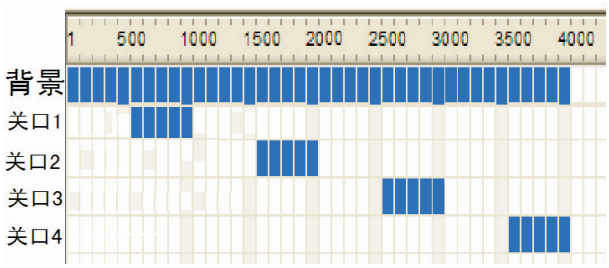

图 4 物流动画在时间轴中的描述

这个动画完整地展示了运煤车从进入卸煤地点到出卸煤地点整个过程中有关进门,检测,称重,卸货,再称重,离开的全部过程,这里利用关键帧设置,使得汽车前进动画持续时间长,而在每一个检测处的停留时间短,动画效果流畅,制作过程简单。

4 结束语

综上完成了一个动画框架该有的基本功能,并使用该引擎制作了几个简单的小动画。该框架分层构建框架的底层程序,具有较强的重用性和可维护性。它具有高度的可扩展性,当用户所需的运动方式或者滤镜效果没有被定义时,用户只需要给出简单的计算公式,即可添加想要的结果。文中由于篇幅有限,部分程序代码没有给出。

参考文献:

[1] 王建民,郑子彬,麦章灿,等. 一个交互自适应手机游戏引擎的设计与实现[J]. 系统仿真学报,2009(10):3120-3122.

[2] 叶 绿. 一个建立在 JXTA 平台上的对等网络游戏框架的设计[J]. 计算机工程与应用,2004,40(17):144-147.

[3] 赵丽娟,朱全银,张 帅,等. 基于 J2ME 的移动网络游戏设计与实现[J]. 计算机工程与设计,2010,31(12):2720-2725.

[4] 冯科融,王和兴. HTML5 网页游戏分析[J]. 网络与通信,2012(24):71-72.

[5] 刘华星,杨 庚. HTML5-下一代 Web 开发标准研究[J]. 计算机技术与发展,2011,21(8):54-58.

[6] 宋 宇,谭 浩. 游戏引擎开发技术简析[J]. 福建电脑,2007(8):31-32.

[7] 成迟薏,石教英,徐迎庆,等. 基于物理模型的窗帘运动实时动画[J]. 软件学报,2000,11(9):1228-1236.

[8] 牛红攀,高 勇,侯忠明. 图形引擎与物理引擎结合的研究与实现[J]. 计算机仿真,2011,28(6):299-303.

[9] Lubbers P,Albers B,Salim F. HTML5 高级程序设计[M]. 北京:人民邮电出版社,2011.

[10] Freeman E,Robson E. Head first HTML5 programming[M]. USA:O'Reilly Media,2011.

[11] Grady M. Functional programming using JavaScript and the HTML5 canvas element[J]. Journal of computing sciences in colleges,2010,26:97-105.

[12] Fulton S, Fulton J. HTML5 Canvas[M]. USA: O'Reilly Media,2011.

(上接第 4 页)

[5] Fritz M H,Leinonen R,Cochrane G,et al. Efficient storage of high throughput DNA sequencing data using reference-based compression[J]. Genome research,2011,21(5):734-740.

[6] 方小永,骆志刚. DNA 序列拼接的分布式并行处理[J]. 计算机工程与科学,2005,27(2):71-73.

[7] Grumbachand S,Tahi F. A new challenge for compression algorithms: Genetic sequences[J]. Information processing & management,1994,30(6):875-886.

[8] Chen X,Li M,Ma B,et al. DNACompress: Fast and effective DNA sequence compression[J]. Bioinformatics, 2002, 18(12):1696-1698.

[9] Rajarajeswari P,Apparao A. DNABIT compress-genome compression algorithm[J]. Bioinformation,2011,5(8):350-360.

[10] Matsumoto T,Sadakane K,Imai H. Biological sequence compression algorithms[J]. Genome informatics series,2000,11:43-52.

[11] 纪 震,周家锐,姜 来,等. DNA 序列数据压缩技术综述[J]. 电子学报,2010(5):1113-1121.

[12] 郑翠芳. 几种常用无损数据压缩算法研究[J]. 计算机技术与发展,2011,21(9):73-76.

超长DNA序列的高效压缩算法研究

作者： 欧阳继超， 冯萍， 康继昌， OUYANG Ji-chao， FENG Ping， KANG Ji-chang
作者单位： 西北工业大学 计算机学院,陕西 西安,710072

刊名： 计算机技术与发展

ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2013(12)