第 35 卷 第 6 期 计 算 机 工 程 2009 年 3 月
Vol.35 No.6 Computer Engineering March 2009
·软件技术与数据库· 文章编号：1000—3428(2009)06—0085—03 文献标识码：A 中图分类号：TP311

基于重叠分块的 FM-index 性能研究与分析

梁 军[1]，张 迪[2]，张云泉[2]
(1. 北京联合大学电子信息实训基地，北京 100101；2. 中国科学院软件研究所，北京 100190)

摘 要：FM-index 压缩查询方法结合压缩技术和索引技术，其最大优势是能够在不解开压缩文件的情况下对源文件进行查询。该文通过理论分析和详细的测试数据研究重叠分块对 FM-index 压缩查询性能的改善。对改进后的 FM-index 和压缩软件 Winrar 在压缩时间和文件占用空间上进行分析和比较，进一步证明了重叠分块对 FM-index 压缩查询性能的改善。

关键词：FM-index 算法；重叠分块；压缩查询

Research and Analysis of FM-index Performance Based on Overlap Blocks

LIANG Jun[1], ZHANG Di[2], ZHANG Yun-quan[2]
(1. Training Center of Electronic Information, Beijing Union University, Beijing 100101;
2. Institute of Software, Chinese Academy of Sciences, Beijing 100190)

【Abstract】FM-index is an advanced compressing information index method in the world. It is an integration of compressing technology and index technology, which supports enquiry of source files under the uncompressing circumstance. This paper researches FM-index compressing in blocks, focuses on the improvement of the capability. Through theoretical analysis and detailed data, analysis and comparisons are made regarding the time and space that FM-index and Winrar take respectively, which demonstrates the improvement of FM-index method.

【Key words】FM-index algorithm; overlap blocks; compressing index

1 概述

压缩查询是压缩技术和索引技术的结合，其目的是支持在不解开压缩文件的情况下对源文件进行查询，因此，它需要特别设计的压缩算法、格式和数据结构来支持压缩状态下的查询。压缩查询是一个很新颖也很有前景的研究领域，压缩查询软件 FM-Index 是目前该领域的佼佼者。

FM-index 是 Ferragina 和 Manzini 设计的压缩查询索引[1]，由一个结合压缩和索引的算法生成，该算法可以像常用的压缩软件那样作为压缩工具使用，同时生成的压缩文件又可以作为信息检索的索引。人们只需要查看压缩文件的一小部分，就可以对源文件的模式进行计数或定位。这种查询对于数兆的文件只需要数微秒的时间[2]。

1.1 FM-index 压缩原理

FM-Index 结构由 2 个部分构成：包含基本数据的核心块和包含支持定位操作的辅助信息块。增加核心块或辅助块的大小都可以加速计数和定位操作。

对于要压缩的文件，FM-index 首先进行 BW 变换，对变换后得到的序列 L 进行 MTF(Move-To-Front)编码，对 MTF 编码后得到的序列 $mtf(L)$再进行一次 RLE 变换，最后将 RLE 变换的结果进行压缩。整个压缩算法称为 BW_RLX 算法。

由于 BW 变换后各算法的目的是将索引尽可能压缩，与查询无关，因此本文重点介绍 BW 变换以及它是如何支持建立索引和查询的。

FM-index 算法是基于 BW 压缩算法和后缀数组数据结构的关联。为了使空间占用率达到理论上的极小值，FM-index 是一种利用压缩索引数据的压缩后缀数组。精确地说，给定文本 $T[1,n]$进行索引，使用 FM-index 最多占用 $5nH_k(T)+o(n)$ 比特的存储空间，其中，$H_k(T)$ 是 T 的 k -th order 熵，并允许在时间 $O(p+occ\log^{1+\varepsilon}n)$ 内查找 $P[1,p]$是否出现在 T 中，$\varepsilon>0$ 是预先设定的任意常量。

BW 变换产生一个原文本的排列，可表示为 $T^{bwt}=bwt(T)$。字符串 T^{bwt} 是按下列步骤得到的结果：(1)添加特殊字符#作为结束标志，将它添加到 T 后面，并定义它为按字典顺序中最小的字符；(2)对文本 T#作循环，然后按字典顺序排序形成概念矩阵 $\boldsymbol{M}$；(3)通过选择 $\boldsymbol{M}$ 的最后一列构造 T^{bwt}。$\boldsymbol{M}$ 的第 1 列被表示为 F。

文本 T 的后缀数组 A 由 T^{bwt} 暗含表示：如果 $\boldsymbol{M}$ 的第 i 行包含字符串 $t_jt_{j+1}\cdots t_n\#t_1\cdots t_{j-1}$，则 $A[i]=j$，FM-index 的这个想法使压缩格式中只需存储 T^{bwt}，然后模拟在后缀数组中查找。FM-index 也可以找出文本中 P 出现的位置，并显示出在其旁边任意长度子串。查找算法的详细说明可参考文献[3-5]。

1.2 基于 FM-index 的查询定位

FM-index 用的是后向检索方法，如在 BWT 变换后的 F=#_ _aaaaabbdllrr 和 L=dra_ll#_aaraabb 中尝试查找 ala。

(1)查询最后一个字符 a，这比较容易，只要在 L 中找到第 1 个和最后一个 a 且映射到 F 中，就可以找到所有的 a。

基金项目：国家审计署与中国科学院合作研究基金资助项目“联网审计技术研究与应用”(KSH1-02)

作者简介：梁 军(1962－)，女，讲师，主研方向：信息检索，并行计算；张 迪，硕士研究生；张云泉，研究员

收稿日期：2008-11-26 **E-mail：**liangjun2236@sina.com

(2)查询倒数第 2 个字符 l。现在可以在 F 中找到第 1 个及最后一个 l 的范围，映射到 F 中，因为所找的 l 肯定在最后一个 a 之前且紧挨着。由于在 L 和 F 中同行的字符是紧挨着的，而且 L 中的在 F 中的同行字符之前，因此可以从 F 映射回 L 从而进一步确定 a 的范围。这里不是通过将所有 L 中的字符扫描一遍看有没有 a，而是只确定一个范围，即从头和尾找 2 次，找到 a 就停止。

(3)由于 a 在 L 中的顺序和在 F 中的顺序是一样的，因此可以由 a 在 L 中是最后 2 个，映射到 F 中也为最后 2 个，此时已找到 2 个 ala 的位置所在，完成了初步的查询定位检索。

2 FM-index 压缩测试与分析

现有的 FM-index 建立压缩索引占用内存过大，从而影响了它的使用和推广，表 1 为对 Proteins.67(67 MB)数据分别用 FM-index 和 Winara 进行压缩后的实验数据。

表 1 FM-index 与 Winrar 压缩测试结果

数据/MB	压缩时间/s		压缩比	
	(FM-index)	(Winrar)	(FM-index)	(Winrar)
10	11	11	0.53	0.34
20	23	23	0.51	0.35
30	45	37	0.51	0.33
40	409	50	0.51	0.34
50	1645	62	0.51	0.34
60	2634	75	0.51	0.34
70	4214	87	0.50	0.37

从表 1 可以看出：(1)FM-index 在压缩比上略高于 Winrar 且压缩比较稳定。(2)当数据大于 30 MB 时，FM-index 在压缩时间上基本与 Winrar 相同，但当数据大于 30 MB 时，时间明显增长，并且不跟随数据呈线性增长，几乎呈平方翻倍，当数据更大时，估计已无法完成，后面进行的另一组实验已证明(即所处理的数据有上限)。而 Winrar 的压缩时间随着数据量的线性增长而增长。

由此得出结论：FM-index 建立压缩索引文件时内存消耗太大，当内存无法满足时就不能进行，即它所能处理的数据有上限，必须加以改进。

3 重叠分块的 FM-index 性能

针对 FM-index 建立索引时大数量级的文本数据存在内存占用过大的问题，本文作了如下的探讨：由于内存有限，一次性将文本读入内存并进行构建，对于大文本而言并不现实。例如，目前普通用户的内存一般不超过 2 GB，使得以目前的 FM-index 版本建立压缩索引的文件限制在 400 MB 以下，该量级对于当前的数据量级来说非常小。因此，内存需求往往成为 FM-index 最大的问题，只有解决了该问题，使 FM-Index 能为 GB 级以上的大文件甚至海量数据建立索引，才能突显其在不解开压缩文件的情况下能进行查询的优势，从而得到推广。

一般情况下，使用 FM-index Version 2 压缩文件需要的内存量为源文件大小的 4 倍~5 倍。因此，结合本文的实验，发现当数量级较小时，FM-index 无论是压缩时间还是压缩比都可以和流行的压缩器媲美，又可以实现不解压状态下的查询。本文提出了分块的 FM-index[6]。相对于初始的 FM-index，进一步把待处理的文本数据划分成块，在每块数据上应用 FM-index 建立独立的索引。这样有以下 4 点好处：

(1)增强数据的局部性。当利用 FM-index 查询文本时需要解压单个的桶，如果使用小块的数据来查询，更有可能使不同的查询结果同时出现在单个的桶中，从而减少了解压桶的次数，提高了查询效率。

(2)减少内存占用。把数据分块以后，每次只处理一块数据，相对于原始的处理，占用的资源肯定减少，可以使用配置较低的机器处理大量的数据。对于内存来说，初始内存占用与数据量的大小呈正比，所以，分块以后可以线性地减少内存占用。

(3)有可能获得时间上的收益。如果数据长度为 L，分块长度为 K，数据处理算法的复杂度为 $f(L)$，则分块后的算法复杂度为$(L/K)\times f(K)$。这可以分 3 种情况讨论。如果算法复杂度是超线性的，则分块后总的复杂度降低，此时在合理的范围内尽量降低分块的大小会大大降低分块后算法的复杂度；如果算法的复杂度是线性的，分块后算法的复杂度不会变化；如果算法复杂度是次线性的，分块后的算法复杂度反而增加，在极端的情况下，如果算法复杂度为常数 $O(1)$，则分块后算法复杂度变为 $O(L/K)$，这时应减少分块的数目以降低分块带来的复杂度。由于数据处理往往是计算密集的，因此分块往往是一种有效的处理方法。

(4)进一步并行化处理。数据分块也是一种常用的并行化方法，但将数据分块后查询模式字符串时，子串可能跨越相邻的 2 个数据块，也就是位于数据块的边界，如果不做任何处理直接查询，这些匹配的结果都会丢失。有 2 种方法来解决这个问题：

1)把子串模式划分成 2 个子串，比如子串模式“ abcde ”可以划分成“ abc ”和“ de ”，然后分别在相邻 2 个数据块的尾部和头部查询划分后的这 2 个子串，如果这 2 个子串同时找到，则可以组合成一个结果，对所有可能的划分都进行查找，然后把结果汇总起来。使用这种方法查询会使计算复杂度明显增加，并且所有相邻 2 个数据块都要相互配合，这样无疑会增加通信开销，使总体性能下降。

2)让 2 个相邻数据块的边缘重叠，在查询子串模式跨越 2 个数据块时，只使用一个数据块就可查询到[8]。这种方法可能出现查询结果重复的现象，需要做一些额外的处理才能最终得到正确的结果。这种方法还要消耗一些额外的空间，但只要选取合适的长度，相对于数据占用的空间这些额外开销可以忽略不计。

本文选用第 2 种方法，提出了如图 1 所示的重叠式分块方法来构建 FM-index，并对其性能进行了测试。通过分块的方法，可以简单有效地提高 FM-index 原型的构建时间和空间性能，并能正确地执行任意的查询操作。具体而言，可以通过损失一些压缩比，将 FM-index 的构建时间从 $O(n\log n)$降到线性，同时使建构空间从线形降到常数。

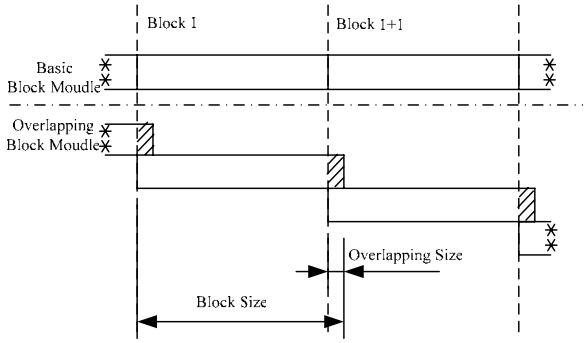


图 1 重叠分块模型

对 test.txt(1.18 GB)数据分别用重叠分块后 FM-index 和 Winara 进行了压缩测试实验，部分结果见图 2。

文件大小/MB	FM-index 分块压缩		FM-index 不分块压缩		FM-index 解压缩	Winrar 压缩		Winrar 解压缩
	时间/s	文件大小/MB	时间/s	文件大小/MB		时间/s	文件大小/MB	
10	4	2.8	4	2.8	2	2	2.7	0.3
20	7	5.7	9	5.7	4	4	4.1	0.5
30	11	8.6	13	8.6	7	6	8.1	0.8
40	16	11.4	17	11.4	10	8	10.7	1.0
50	18	14.2	23	14.2	13	10	13.4	1.0
60	22	17.0	28	17	16	12	16.1	2.0
70	27	19.8	33	19.8	19	14	18.7	2.0
80	30	22.6	38	22.6	23	16	21.4	3.0
90	34	25.5	41	25.5	25	18	22.5	3.0
100	38	28.3	47	28.3	28	21	26.8	3.0
110	42	31.1	54	31.1	33	23	29.5	3.0
120	45	33.9	58	33.9	36	25	32.2	3.0
130	49	36.7	64	36.7	39	27	34.9	4.0
140	52	39.5	71	39.5	44	29	37.6	4.0
150	56	42.2	79	42.2	49	31	40.2	4.0
160	60	45	88	45.0	53	33	42.9	4.0
170	64	47.8	110	47.8	61	35	45.3	5.0
180	67	50.6	137	50.6	110	37	48.3	5.0
190	71	53.2	205	53.2	系统内存不支持	39	49.2	5.0
200	75	56.7	内存不支持			42	53.6	6.0

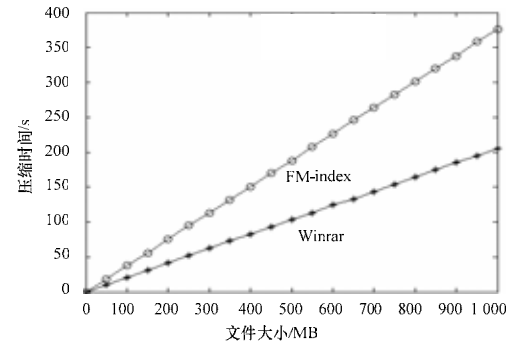
图 2　重叠分块后的 FM-index 与 Winrar 压缩测试结果

通过以上对高数量级数据(1.18 GB)的测试结果表明，分块构建的方法明显改善了压缩性能。

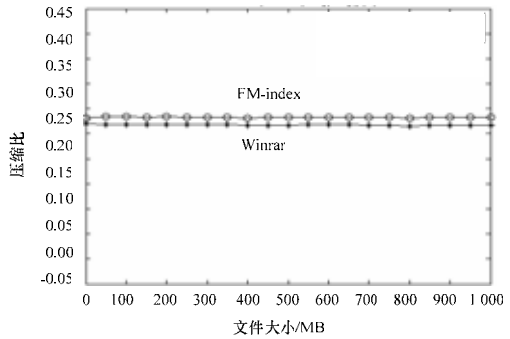
(1)从图 3 中可以看出，分块后 FM-index 的压缩比只是略高于 Winrar 且很稳定。

(2)分块后 FM-index 的压缩时间随文件大小的变化而呈线性变化，因为要构建索引，所以压缩时间略高于 Winrar。

(3)从图 4~图 7 的分析可看出，FM-index 的优势在于可实现不解压状态下的查询且节省存储空间。



(a)压缩时间



(b)压缩比

图 3　基于分块的 FM-index 与 Winrar 压缩测试数据分析

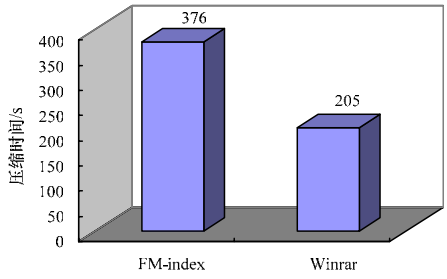


图 4　数据为 500 MB 时压缩时间

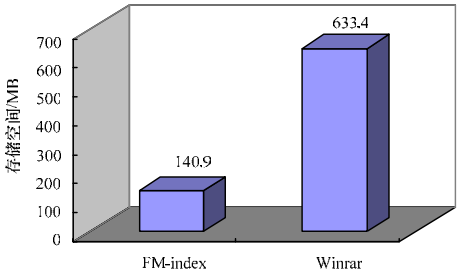


图 5　数据为 500 MB 时空间对比

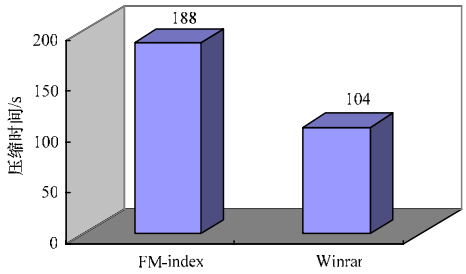


图 6　数据为 1 000 MB 时压缩时间

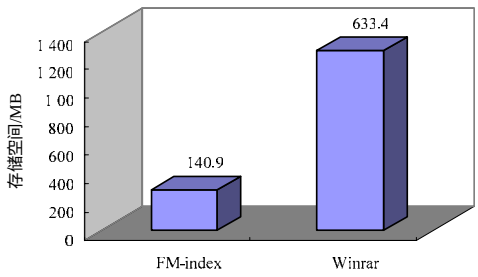


图 7　数据为 1 000 MB 时空间对比

4　结束语

本文研究了 FM-index，在压缩时间和压缩空间上与现有压缩软件 Winrar 进行了分析和比较。设计一种基于重叠分块处理的方法构造 FM-index。它能在线性时间内使用常量级临时的内存空间建立 FM-index，特别适合处理大数量级文件。相对于把装载和添加索引作为一个整体，这种新的方法把文件分成固定大小的各个子块，然后分别添加索引。为了保证查询操作的正确性和高效性，在添加索引前，进一步把后续字符的确切长度添加到每个子块的最后。

实验结果表明，基于重叠分块的 FM-index 压缩性能有了很好的改善，使压缩速度与文件大小无关，只与分块数据的大小有关。而且 FM-index 的优势在于可实现不解压状态下的查询且节省存储空间，具有良好的应用前景。笔者在 Matlab 软件中分析了 FM-index 和 Winrar 在不同平台上进行数据压缩时的变化规律，为此类课题的研究提供了参考依据。

(下转第 93 页)