基于SSE指令集的串匹配算法优化

邵妍[1,2,4]，刘燕兵[1,3,4]，刘萍[1,4]，郭莉[1,4]

（1. 中国科学院计算技术研究所，北京 100190；2. 北京邮电大学，北京 100876；

3. 中国科学院研究生院，北京 100039；信息内容安全技术国家工程实验室，北京 100190）；

摘　要：串匹配是计算机研究领域的经典问题之一，在网络安全、计算生物学、信息检索等领域发挥着关键的作用。其中，基于位并行的串匹配算法所需存储空间小、匹配速度快，但由于受到机器字的限制，只适合小规模的串匹配。基于SSE系列指令集对经典的位并行算法Shift-And、BNDM进行了优化，优化算法利用SSE指令集提供的128位大位宽寄存器，将多个状态向量打包到SSE寄存器上，并通过SSE的位操作指令状态向量进行更新。在随机数据和真实数据上的测试结果显示，优化算法的匹配速度达到原算法的2倍以上。

关键词：串匹配算法；位并行；SSE指令集；Shift-And；BNDM

1 引言

精确串匹配（后面简称“串匹配”）问题是计算机科学研究领域的一个经典问题。它指在文本 $T=t_1t_2\ldots t_n$ 中找出某个给定的模式串集合 $P=\{p_1,p_2,\ldots,p_r\}$ 的所有出现位置，其中 T 和 $p_i(1\leqslant i\leqslant r)$ 是在有限字符表 Σ 上的字符序列。串匹配技术广泛应用于网络信息安全领域，典型的应用包括：入侵检测/防御系统（IDS/IPS）、反病毒和反垃圾邮件检测（AV/AS）、网络带宽管理和服务质量（QoS）、统一威胁管理（UTM）等。

近年来，随着宽带技术的发展和多媒体应用的广泛流行，互联网技术得到极大的普及和发展。由于互联网协议和计算机系统的漏洞，网络安全形势日趋严峻。传统的网络内容安全处理技术的存储空间和运算速度已经难以满足实时高速网络环境下的应用需求，面向高速网络内容安全处理的串匹配技术已经得到学术界和工业界的广泛关注，成为网络信息安全领域的研究热点。

SSE系列指令集是Intel为提高流媒体处理性能而推出的多媒体向量指令集，包括SSE、SSE2、SSE3、SSSE3、SSE4，目前已更新到SSE4.2指令集。SSE系列指令集定义了8个全新的128位寄存器，支持在128位数据的并行操作。从理论上来说，SSE指令的处理速度可以达到普通指令的4倍。如今，SSE指令集在多媒体处理领域已经得到广泛应用，如图像处理、视频处理、音频处理等。在信息加密和科学计算方面，SSE指令集也在发挥着作用。鉴于SSE指令集在数据并行处理上的高效性能，我们将把SSE指令应用到串匹配领域，利用SSE指令集提供的大位宽寄存器对串匹配算法进行优化。

本文基于SSE系列指令集，选取合适的指令，对经典的位并行串匹配算法Shift-And和BNDM进行了优化。实验结果显示：通过SSE的优化，算法的匹配速度提高到原始算法的2倍以上。本文以下章节安排如下：第2节对位并行串匹配算法、SSE指令集及其相关指令的测试、利用SSE指令集优化串匹配算法的相关工作进行了介绍；第3节阐述了基于SSE对Shift-And和BNDM进行优化的方法；第4节给出了优化算法在随机数据集和真实数据集上的测试结果；第5节为结论。

2 相关工作

2.1 位并行串匹配算法介绍

串匹配算法的分类有多种，按照其关键技术来分，可以分为以下三类：

(1) 基于自动机的匹配算法，如Aho-Corasick[1]算法、BOM[2]算法。这类算法的基本思想是：将模式串集合以自动机形式存储，匹配过程就是状态转移过程。它的优点是性能稳

基金项目：国家973计划重点基础研究项目（2007CB311100）；国家242信息安全计划项目（（242）2009A43）

定，使用范围广；缺点是所需存储空间大，匹配速度较慢。

(2) 基于散列的匹配算法，如 Wu-Manber [3] 算法、Karp-Rabin[4]算法。这类算法的基本思想是：将模式串集合以散列表的形式存储，匹配过程就是查表的过程。它的优点是所需存储空间小，匹配速度快；缺点是对环境敏感，性能不稳定。

(3) 基于位并行的算法，如 Shift-AND/OR[5] 算法、BNDM[6]算法。这类算法的基本思想是：将模式串集合与文本串的匹配状态用位向量存储，匹配过程就是用位操作更新位向量的过程。它的优点是所需存储空间小，匹配速度快；缺点是算法性能会随模式串个数的增多而下降，只适合中小规模的模式串集合。

SSE 指令集适合应用于并行度高、耦合度小、判断分支少的程序和数据。基于自动机的算法依赖自动机的状态转移，判断分支非常多，不符合 SSE 指令优化的特点。基于散列的匹配算法在匹配过程中主要进行散列函数的递归计算，并行度低，且 SSE 指令集中没有专门针对此类计算的指令，也不适于用 SSE 指令进行优化。基于位并行的算法匹配过程完全用位向量和位操作来完成，具有天然的并行性，适合用 SSE 指令进行优化。因此，我们主要针对基于位并行的这类串匹配算法进行基于 SSE 指令集的优化。

Shift-And 算法和 BNDM 算法是基于位并行的串匹配算法中的典型代表。虽然在具体的匹配方式上有差异，但两种算法的核心思想一致：通过位向量来记录模式串与文本的匹配状态，每读入一个新的文本字符，就通过位并行技术来更新该位向量。下面以 Shift-AND 算法为例进行介绍。

Shift-And 算法维护一个字符串的集合，集合中的每个字符串既是模式串 p 的前缀，同时也是已读入文本的后缀。每读入一个新的文本字符，该算法采用位并行的方法更新该集合，该集合用一个位掩码 $D=d_m...d_1$ 来表示。D 的第 j 位被置为 1，当且仅当 $p_1...p_j$ 是 $t_1...t_i$ 的后缀。

Shift-And 算法首先构造一个 m 位的向量表 $B[\sigma](\sigma\in\Sigma)$，用来记录字符 σ 在模式串的出现位置。如果 $p_j=\sigma$，掩码 $B[\sigma]$ 的第 j 位被置为 1，否则为 0。首先置 $D=0^m$，对于每个新读入的文本字符 t_{i+1}，用如下公式对 D 进行更新：$D[i+1]=((D[i]<<1)|0^{m+1}1)\&B[t_{i+1}]$。在匹配时，逐个扫描文本字符并更新向量 D，当 $D[i]\&10^{m-1}\neq 0^m$ 时，在文本位置 i 处匹配成功。

Shift-And 算法扩展到多模式串时，将所有模式串的位向量 D 包装到一个机器字里，用位并行技术同时对 r 个位向量进行更新，初始化字和匹配掩码分别是所有初始化字和所有匹配掩码的连接。

设机器字的长度为 w，文本串的长度为 n，模式串的个数为 r，最短模式串长度为 m，那么 Shift-And 算法和 BNDM 算法的时间复杂度为 $O(n\left\lceil\frac{mr}{w}\right\rceil)$。由于采用了位并行技术，Shift-And 算法和 BNDM 算法的匹配速度是很快的。但一旦模式串的长度和超出机器字的长度，两种算法的性能都会发生明显下降。

2.2 SSE 指令集简介及其关键指令测试

截止到目前最新的 SSE4.2 指令集，总共有超过 280 条 SSE 指令。按照功能分类，SSE 指令可分为六类：算术操作、位操作、比较、数据移动、缓存预取、字符串比较。SSE 系列指令集定义了 8 个全新的 128 位寄存器，可以通过名称 XMM0-XMM7 直接访问（如图 1 所示）。XMM 寄存器与浮点寄存器、MMX 寄存器以及通用寄存器完全独立。因此，应用程序可以在进行整数 SIMD 操作（即 MMX 操作）的同时，进行单精度浮点数的 SIMD 操作（SSE）。同样，SSE 还可以在执行浮点数的非 SIMD 操作的同时，进行 SIMD 操作。

SSE 系列指令集采用了 SIMD（Single Instruction Multiple Data）技术，即单指令多数据技术，指用一个控制器对一组数据（又称“数据向量”）中的每一个分量分别执行相同的操作，从而实现空间上的并行性。SSE 寄存器位宽 128 位，相当于普通 32 位机器位宽的 4 倍，可同时容纳 4 个 32 位整数。在 SSE 寄存器的支持下，SSE 指令可以同时对 4 个 32 位整数进行并行操作。从理论上来说，SSE 指令集较普通指令的加速比可以达到 4 倍。

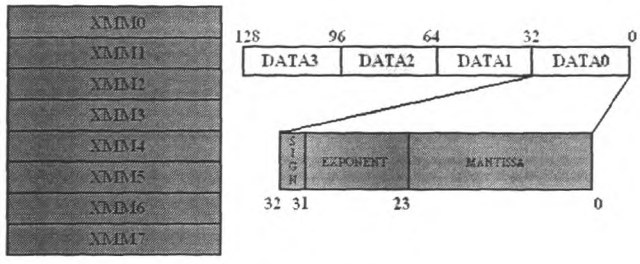

图1 SSE寄存器示意图

Shift-And算法和BNDM算法在匹配过程中，频繁地对位向量进行位操作。因此，我们通过内嵌原语的方式对SSE位操作指令进行了测试，包括128位的逻辑与运算、128位的逻辑或运算、64位的左移运算和128位的位比较指令。选取的SSE位操作指令的说明如表1所示。由于SSE的128位移位指令要求移位的位数必须是8的整数倍，而串匹配算法中通常需要1位的移动操作，128位移位指令不能满足需要。因此，我们选择使用64位移位指令并对其进行测试，该指令将128位的操作数截断为2个64位的整数，分别进行移位。

完成测试的实验环境如下：CPU：Intel Xeon E5520，2.27GHz，2路4核；内存：8GB；Cache：32KB L1数据cache，32KB L1指令cache，256KB L2 cache，8MB共享L3 cache；操作系统：Windows 7 Ultimate，64位系统；编译平台：Visual Studio 2010。

表1测试的SSE位操作指令说明

操作类型	原型	返回值说明	所属指令集
逻辑与	__m128i _mm_and_si128(__m128i a,__m128i b)	r:=a&b	SSE2
逻辑或	__m128i _mm_or_si128 (__m128i a, __m128i b)	r:=a|b	SSE2
左移	__m128i _mm_slli_epi64(__m128i a, int count)	r_0:=a_0<<count r_1:=a_1<<count	SSE2
位比较	int _mm_testc_si128(__m128i a, __m128i b)	r:=((a&b)==b)	SSE4.1

我们将不同类型的位操作分别重复执行10000次，得到的普通指令和SSE指令操作用时如图2所示。实验结果显示：逻辑与操作上，SSE指令的速度是普通指令的2.29倍；逻辑或操作上，SSE指令的速度是普通指令的2.39倍；左移操作上，SSE指令的速度是普通指令的1.37倍；位比较操作上，SSE指令的速度是普通指令的3.92倍。可见，SSE指令较普通指令有较明显的性能提升，将SSE指令应用到Shift-And和BNDM中，将对算法的性能起到提升作用。

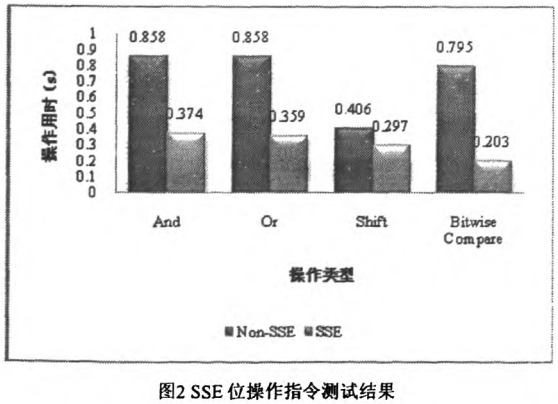

图2 SSE位操作指令测试结果

2.3 利用SSE指令对串匹配算法进行优化的相关工作

尽管SSE指令集在多媒体处理领域已经得到广泛的应用，但是，目前在SSE应用于串匹配算法方面进行的研究还比较少。近年来，国内外对SSE指令集应用在串匹配领域的研究方向主要有两种：一种是针对SSE指令集的具体指令设计和实现精确串匹配算法；另一种是利用SSE指令集并行计算的特点来优化计算生物学中的近似串匹配算法。

在精确串匹配方面，文献[7]对朴素串匹配算法进行了基于SSE2的优化，主要思想是：每次从目标串中取出16字节作为一个整体，利用SSE2指令，与模式串进行匹配。优化算法平均情况下性能提高41%，只支持单串匹配算法。

文献[8]设计了一种单模式串的过滤算法，该算法通过比较模式串和带匹配文本的哈希值进行过滤，在计算哈希值时用到SSE指令，分别为64位的左移指令_mm_slli_epi64和符号提取指令_mm_movemask_epi8。该算法性能有40%-50%的提升，只适用于长度大于32字节的长字符串。

在近似串匹配方面，文献[9]提出了一种基于SIMD技术的Smith-Waterman算法的改进算法SWMMX。SWMMX算法中将F的计算创造性地进行分解，使其元素之间的相互依赖转变为对相

应左上元素的依赖，并利用 SWAT 优化从而快速计算出矩阵 F 各单元的值。作者用 MMX 和 SSE 技术对算法进行了实现，在序列相似性显著的情形下，其性能六倍于 ssearch[10]。

文献[11][12]也是基于 SSE 指令集对 Smith-Waterman 算法的改进。文献[11]从简化数据依赖关系出发，采用前驱计算思想，改进了 Smith-Waterman 算法得分矩阵 H 中 F 的计算，并采用 SSE2 指令集进行了实现。在相似性显著的情况下比普通的 SW 算法性能提高 5 倍，且与测试集无关。文献[12]提出一种对 Smith-Waterman 算法的不同实现，利用 SSE2 指令进行加速。

文献[13]提出了一种蛋白质序列的快速模式匹配方法，该算法基于 SSE 指令集对 BALST 算法进行优化，采用 L1 范数度量 k-元组。该算法适合采用 SIMD 指令进行加速，适用于近似匹配问题。

文献[14]提出了一种适用于 DNA 小字符集的碱基序列近似匹配算法，用 MMX 指令集提供的 64 位寄存器和逻辑运算指令进行了高效实现，性能有 20%的提升。

3.Shift-And 和 BNDM 基于 SSE 系列指令集的优化实现

上一节中分析了 Shift-And 和 BNDM 的时间复杂度，机器字的长度越长，算法的时间复杂度越低。因此，我们从拓宽操作字长度出发，利用 SSE 指令集提供的超大位宽寄存器对 Shift-And 算法和 BNDM 算法的性能进行优化。

3.1 优化思想

SSE 指令集提供了一组大位宽的寄存器，支持 128 位数据的并行操作。我们基于 SSE 指令集对 Shift-And 和 BNDM 进行了优化，优化后的算法分别称为 SSE-Shift-And（也简写为 SSE-SA）和 SSE-BNDM。优化算法通过利用 SSE 指令拓宽了原算法操作字的长度，使优化后的算法支持更大规模的模式串匹配。

对Shift-And算法和BNDM算法进行SSE优化的基本思想是：采用位向量D记录每个模式串前缀或子串和当前文本t的匹配状态，将多个模式串的状态向量打包（pack）到 128 位的SSE寄存器上（如 图 3所示）。在匹配的过程中，每读入一个文本串字符就通过SSE指令集提供的位操作指令（逻辑或、逻辑与、左移）更新状态向量D，并通过SSE指令集提供的位比较指令来检测特定的位以判断模式串和文本是否匹配成功。

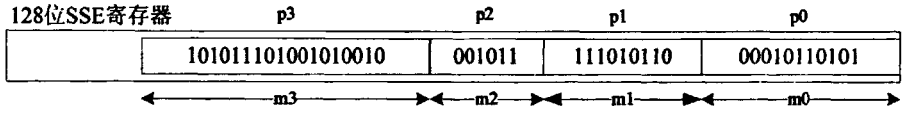

图3 SSE-Shift-And 和 SSE-BNDM 基本思想示意图

3.2 优化算法的实现

表2 Shift-And 和 BNDM 算法 SSE 优化的指令选取

操作类型	选取的 SSE 指令		
	SSE 指令原型	返回值说明	所属指令集
逻辑与	__m128i _mm_and_si128 (__m128i a, __m128i b)	r:=a&b	SSE2
逻辑或	__m128i _mm_or_si128 (__m128i a, __m128i b)	r:=a|b	SSE2
左移	__m128i _mm_slli_epi64(__m128i a, int count)	r0:=a0<<count r1:=a1<<count	SSE2
位比较	int _mm_testz_si128(__m128i a, __m128i b)	r:=((a&b)==0)	SSE4.1
	int _mm_testc_si128(__m128i a, __m128i b)	r:=((a&b)==b)	SSE4.1

对Shift-And算法和BNDM算法进行优化时，主要用到的位操作包括逻辑与、逻辑或、左移和位比较。在优化算法的实现中，选用的SSE指令的说明如 表 2所示。

在算法的预处理阶段，计算表示字符σ在字符串中出现位置的表$B[\sigma]$时，通过对一个最低位为 1、其他位均为 0 的__m128i型数据进行左移来实现（如所 图 4示）。但由于 64 位左移指令 _mm_slli_si128 将 128 位的整数截断为 2 个 64 位的整数，最低位上的 1 无法从低端 64 位穿过 图

4中所示的粗线移动到高端的64位上来。因此，我们定义了两个__m128i类型的数据，分别命名为onel和oneh，onel如图4所示，oneh如图5所示。计算表$B[\sigma]$的过程中，当左移位数小于等于63位时，用onel进行左移；当左移位数大于63时，用oneh进行左移。

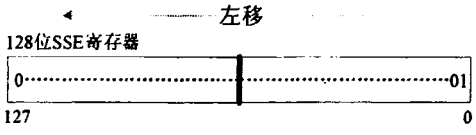

图4 预处理阶段计算表B的示意图及定义的onel

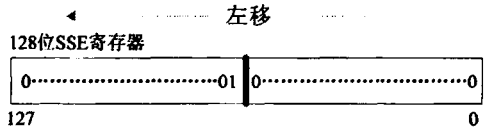

图5 预处理阶段为计算表B定义的oneh

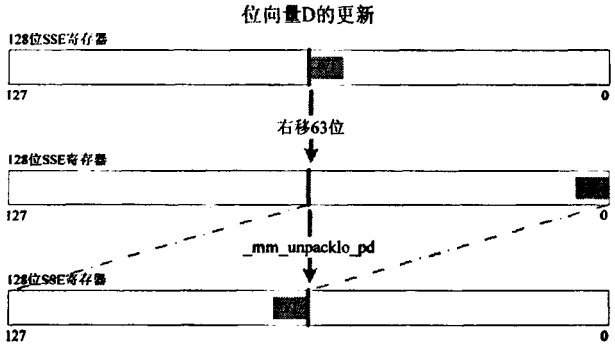

图6 向量D更新操作的示意

在匹配阶段，更新记录模式串前缀或子串与当前文本匹配状态的位向量D时，需要对位向量D进行左移一位的操作。此时，同样是由于64位左移指令_mm_slli_si128对128位的整数的截断，63位上的数据信息左移一位后会被丢失掉。为解决这个问题（如图6），首先将位向量D右移63位，将63位的数据信息移到最低位。然后，采用SSE指令集中的指令_mm_unpacklo_pd，这条指令返回两个操作数低端的64位，指令原型为：

__m128d _mm_unpacklo_pd (__m128d a, __m128d b)

Return Value r0 := a0 r1 := b0

这样，就将左移移位前63位上的数据信息保留下来，并移到了64位。

4.实验

本节对优化算法SSE-Shift-And、SSE-BNDM和原始算法Shift-And、BNDM的性能分别进行了对比测试，测试数据包括随机数据和真实数据两类。我们将优化算法的匹配速度与原算法的匹配速度的比值称为加速比。采用的实验环境如下：CPU：Intel Core 2 Duo E8400 3.00GHz；内存：2GB DDR2；Cache：32KB L1数据cache，32KB L1指令cache，6144KB L2 cache；操作系统：Windows 7旗舰版；编译平台：Visual Studio 2008。

4.1 随机数据下的测试结果

在随机数据测试中，字符集大小为256，模式串和待扫描的文本数据都随机生成，待测试的文本数据大小为10MB。

第一个实验的目的是测试算法的加速比与模式串个数的关系。实验中，模式串长度为6B，模式串规模为100-1000，以100递增。SSE-Shift-And算法和SSE-BNDM算法的加速比如图7所示。SSE-Shift-And算法的速度保持在Shift-And算法的2.6-3.2倍，SSE-BNDM算法的速度保持在BNDM算法的2.1-2.6倍。可以看出：随着模式串个数的增加，两种算法的加速比基本保持不变。

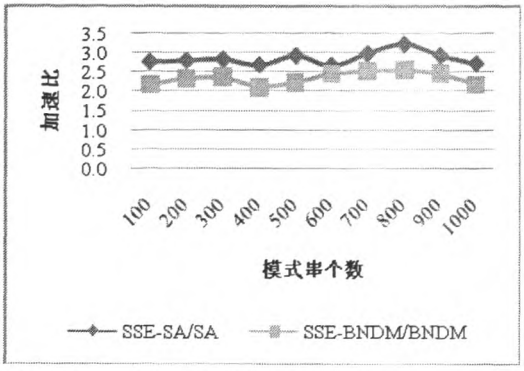

图7 优化算法和原始算法的速度比值（模式串长度为 6B）

第二个实验的目的是测试算法的加速比与模式串长度的关系。实验中，模式串规模为 10，模式串长度 2B-20B，以 2B递增。SSE-Shift-And算法和SSE-BNDM算法的加速比如 图 8所示。SSE-Shift-And算法的速度从Shift-And算法的 0.8 倍逐渐上升到 2.7 倍，SSE-BNDM算法的速度从BNDM算法的 0.9 倍逐渐上升到 2.4 倍。可以看出：随着模式串长度的增加，两种算法的加速比逐渐增大，并有继续上升的趋势。

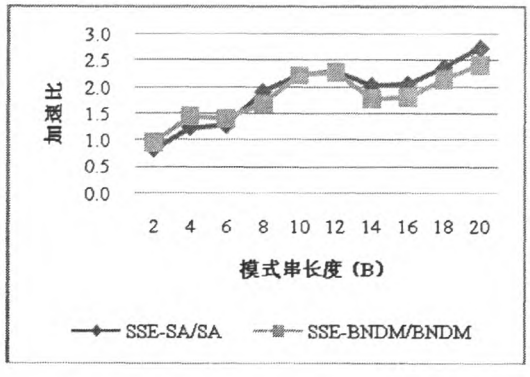

图8 优化算法和原始算法的速度比值（模式串个数为 10）

第三个实验的目的是测试在模式串总长度不变（小于当前的机器字长度）的情况下，算法的加速比与模式串长度的关系。实验中，模式串的长度和为 128B，模式串长度相等，2B-64B。SSE-Shift-And算法和SSE-BNDM算法的加速比如 图 9所示。随着模式串长度的增加，两种算法的加速比逐渐增大。这说明，模式串长度和一定时，模式串长度越长，优化算法的匹配速度优势越显著。

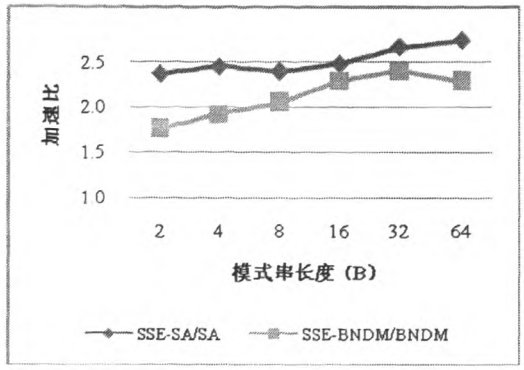

图9 优化算法和原始算法的速度比值（模式串个长度和为 128B）

4.2 真实数据下的测试结果

在这组测试中，我们选择了三类真实数据。对测试数据的说明如下：

1) ClamAV 数据

ClamAntiVirus 是一个开源的防病毒软件，病毒库随时更新，最新版本可在 http://www.clamav.net/binary.html#pagestar 获得。我们使用了 ClamWin Free AntiVirus 0.90.1 版本，因为其中许多病毒签名中包含各种通配符，不适于在精确串匹配中使用。因此，我们提取了签名中最长片断，最小长度为 6 字节，从中随机提取 98 个组成测试用模式串集合。

待扫描的文本数据来自 MIT 公开的一组真实的网络数据，它的原始目的是用来对网络入侵检测系统进行评估，可在 http://www.ll.mit.edu/IST/ideval/下获得。我们使用了 mit_1999_training_week1_friday_inside.dat，因为数据太大会导致匹配时间过长，我们截取了其中的一部分，大约在 10M。

2) DNA 数据

DNA 数据来自于 Ncbi 分子生物学数据库，我们选取了果蝇的 X 染色体，下载地址为 http://www.ncbi.nlm.nih.gov/mapview/map_search.cgi?taxid=7227，文件为 sequences.fasta.gz，长度大约 22M。我们对文件中的 0D 0A 'N'进行了处理，填充了随机的 A T G C 字符。

为了对这种 DNA 数据进行匹配测试，我们书写了特征串抽取程序，从中随机抽取指定长度和个数的字符串作为测试用特征串集合。测试中使用了长度为 30 字节的特征串 50 个。

3) 英文数据

英文数据中的待匹配数据使用的是《圣经》的纯文本文件，大小为 3,953KB，包含约 76 万单

词。

我们用书写的特征串随机抽取程序从中抽取指定个数的单词作为测试用特征串集合。测试中抽取了120个关键词，长度为6-13字节。

ClamAV数据、DNA数据、英文数据的测试结果分别如图10、图11和图12所示。测试结果显示：基于SSE系列指令集的优化算法在三类真实数据上的匹配速度都大于原始算法。通过SSE指令的优化，Shift-And算法和BNDM算法的匹配速度得到了显著的提升。

表3对真实数据上的测试结果进行了更加详细的展示。在内存使用量上，基于SSE指令的优化算法略大于原始算法。SSE-Shift-And算法的内存使用量较Shift-And算法增长不超过10.3%；SSE-BNDM算法的内存使用量较BNDM算法增长不超过5.3%

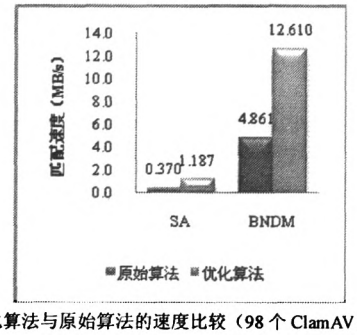

图10 优化算法与原始算法的速度比较（98个ClamAV关键词）

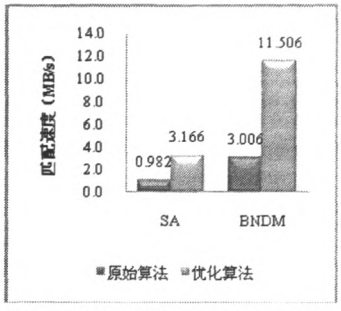

图11 优化算法与原始算法的速度比较（50个DNA关键词）

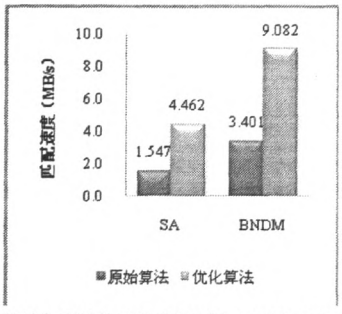

图12 优化算法与原始算法的速度比较（120个bible关键词）

表3 优化算法和原始算法的真实数据测试结果

测试数据	算法	内存使用量（MB）	匹配速度（MB/s）	命中次数	加速比
98个ClamAV关键词	Shift-And	0.124	0.370	2359	3.2
	SSE-Shift-And	0.127	1.187	2359	
	BNDM	0.019	4.861	2359	2.6
	SSE-BNDM	0.020	12.610	2359	
50个DNA关键词	Shift-And	0.046	0.982	50	3.2
	SSE-Shift-And	0.046	3.166	50	
	BNDM	0.047	3.006	50	3.8
	SSE-BNDM	0.047	11.506	50	
120个bible关键词	Shift-And	0.029	1.547	31046	2.9
	SSE-Shift-And	0.032	4.462	31046	
	BNDM	0.023	3.401	31046	2.7
	SSE-BNDM	0.024	9.082	31046	

4.3 实验结果的分析

通过在随机数据和真实数据上的测试，可以看出：经过SSE指令的优化，经典的位并行算法的匹配速度得到了显著的提升，优化效果明显。在不同的数据集合上，优化算法的加速比是相当的，优化后算法的匹配速度基本都提升到原始算法的2-3倍。

从理论上来说，SSE 指令集较普通指令的加速比应该达到 4 倍，但实际上经过 SSE 优化后算法的匹配速度只提升到原始算法的 2-3 倍，与理论上的加速比有一定差距。这主要是两方面的原因造成：一方面，从第二节 SSE 指令的测试结果可以看出，SSE 指令的加速比在 2.3-3.9 倍，达不到理论上的 4 倍；另一方面，更新位向量 D 时，由于 SSE 指令的限制，加入了额外的右移和低位提取操作，对算法的速度也造成了一定影响。

5 结论

本文基于 SSE 系列指令集，对经典的位并行算法 Shift-And、BNDM 进行了优化，取得了明显的效果。在实际工程应用中，这种基于 SSE 的优化可以为系统带来性能的提升，但是对于一个现有的系统而言，这种性能提升是有代价的。从硬件上来说，并不是所有的 CPU 都能提供对 SSE 指令集的支持。现有系统中一些较陈旧的机型可能无法支持较新的 SSE 指令，这就需要更新现有的 CPU，系统改造代价较大。从软件上来说，应用 SSE 指令对系统使用的操作系统和编译器都有一定要求，这就需要对系统运行的操作系统和编译器做相应调整。相对而言，这种改造的代价较小。因此，在使用 SSE 指令对一个现有系统进行改造时，要综合考虑改造带来的收益和付出的代价，改造与否是一个需要权衡的问题。

参考文献

[1]. AHO A. V., CORASICK M. J.. Efficient string matching: an aid to bibliographic search. Communications of the ACM, 1975, 18:333-340.

[2]. CROCHEMORE M., ALLAUZEN C., RAFFINOT M.. Factor oracle: a new structure for pattern matching. Theory and Practice of Informatics, 1999:291-306.

[3]. WU S., MANBER U.. A fast algorithm for multi-pattern searching. Technical Report Department of Computer Science Chung-Cheng University.1994,5.

[4]. KARP R., RABIN M. Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development 31(1987), 1987, 3:249-260.

[5]. BAEZA-YATES R., GONNET G. H.. A new approach to text searching. Communications of ACM, 1992, 35(10):74-82.

[6]. NAVARRO G., RAFFINOT M.. Fast and fexible string matching by combining bit-parallelism and suffix automata. ACM Journal of Experimental Algorithms, 2000, 5(4):1-36.

[7]. 戴正华，徐琳，冯圣中. 基于 SSE2 的高性能串匹配算法研究. 第 8 届全国并行计算大会论文集, 大连, 2004.

[8]. Külekci M. O.. Filter Based Fast Matching of Long Patterns by Using SIMD Instructions. Proceedings of PSC 2009, Czech Republic, 2009: 118-128.

[9]. ROGNES T., SEEBERG E.. Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. Bioinformatics, 2000, 16: 699-706.

[10]. PEARSON W. R.. Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. Genomics, 1991, 11(3): 635-650.

[11]. 戴正华,张庆丹,徐琳等. 基于 SSE2 的 Smith-Waterman 算法. 计算机工程与应用, 2006, 42(11).

[12]. Witold R. Rudnicki, Aleksander Jankowski, Aleksander Modzelewski, Aleksander Piotrowski, Adam Zadrożny. The new SIMD Implementation of the Smith-Waterman Algorithm on Cell Microprocessor. Fundamenta Informaticae .96 (2009).2009: 181-194.

[13]. Méndez J., Falcón A., Lorenzo J.. A Procedure for Biological Sensitive Pattern Matching in Protein Sequences. Lecture Note in Computer Science, 2003, 2652/2003: 547-555.

[14]. KISHIMOTO MAKOTO, KANEKO KEIICHI. A Matching Algorithm for Base Sequences with Bit-Parallel Approach Using MMX Instructions. 2003, 2003(29): 143-148

邵妍（1987-），女，天津人，北京邮电大学硕士生，主要研究方向为字符串处理相关的算法、信息安全等。

刘燕兵（1981-），男，湖北人，博士生，中国科学院计算技术研究所助理研究员，主要研究方向为字符串处理相关的算法、信息安全等。

刘萍（1972-），女，山东人，中国科学院计算技术研究所助理研究员，中国计算机学会会员，主要研究方向包括信息安全，数据流处理，字符串匹配算法。

郭莉（1969-），女，湖南人，中国科学院计算技术研究所正研级高级工程师、硕士生导师，主要研究方向包括信息安全，数据流处理。

基于SSE指令集的串匹配算法优化

作者： 邵妍， 刘燕兵， 刘萍， 郭莉

作者单位： 邵妍(中国科学院计算技术研究所,北京100190;北京邮电大学,北京100876;信息内容安全技术国家工程实验室,北京100190)， 刘燕兵,刘萍(中国科学院计算技术研究所,北京100190;中国科学院研究生院,北京100039;信息内容安全技术国家工程实验室,北京100190)， 郭莉(中国科学院计算技术研究所,北京100190;信息内容安全技术国家工程实验室,北京100190)