# 1.开场

在座的各位老师好，我叫王力，是戴林老师的学生。今天我答辩的题目是SIMD指令在生物序列串匹配里的应用。

首先说一下研究背景。

# 2.研究背景

进入二十一世纪以来，生命科学和生物技术伴随着人类基因组计划的成功实施得到了迅猛发展，基因组学相关数据也呈现指数式增长。比如最早的人类基因组计划，即测出成人染色体中所包含的30亿个碱基对组成的核苷酸序列。与1990年正式启动，多国科学家花费13年，耗资27亿美元才完成。但2007年之后，高通量测序仪器获得不断的改进，现在对个人基因组测序只需5000美元。

由此可见，测序成本的下降导致了生物基因序列信息的迅速扩张，因此对生物信息学研究的基础工具提出了更高的要求，而串匹配算法又是基因序列比对等数据分析计算的基石。同时，由于专门针对生物信息的串匹配算法不多且较为陈旧。因此，找出一个适用于基因序列且更有效率的算法显得十分必要。

# 3.研究现状

早期的串匹配算法只是在经典算法（如KMP，BM，QS，shift-or）的基础上针对特定情形的序列进行了一系列优化。

另一方向是利用GPU的并行计算框架来处理，但多数机器未必能配上对应的版本环境，导致应用场景有限，并且目前的多数改进只是将经典算法用GPU的并行计算框架实现一遍，以此获取性能提升。

还有一个方向是利用Intel提供的SSE指令去改进原有算法，即将利用指令中的128位大寄存器，将多个字符打包成一个较大的字，即块字符，来一次性处理。

# 4.生物序列特性

生物序列字典较小，尤其是基因序列字典为4。实验表明，序列表现与等字典大小的随机串类似。

介绍新算法前，先介绍新算法会使用的几条模拟指令，这些模拟指令均由INTEL SSE指令组成。

# 5.字长比较指令

wscmp指令被设计来比较两个ω位的字，并返回一组α个γ位的值。如图所示。

根据样例中的赋值，假定该指令能一次性处理两个48位的数据（实际情况是能一次性处理两个128位的数据）。

在这条指令中，两个48位的字a和b被分成12组，每组是4位的数值。然后这些数据按序比较得到数值r。假定，b，那么wscmp指令将返回一个12位的值，即，其中当且仅当=时，=0，否则=1。

该伪指令能在一个较短的常数时间内模拟，它由如下两条SSE指令组成。

第一条指令分别比较寄存器\_A和寄存器\_B对应位置8位整数是否相等，若相等，则该位置返回0xff，否则返回0x0。

第二条指令的传参是一个128位的数值，该指令返回一个16位的数值为其签名。

# 6.字长匹配指令

wsmatch指令被设计用来返回短串b在a中的出现次数和位置。

如图所示，短串b在a的1,5位置出现，

假定，b，那么该指令将返回一个12位的数值，即，其中当且仅当对于，有，则=0，否则=1。也就是说只有当短串b出现在字符串a的起始位置i的子串中，那么=0。

该伪指令能在一个较短的常数时间内模拟，它由三条SSE指令组成，后两条即为wscmp指令。

其中，z是一个128位的数值，初始设为。

第一条指令又称为单指令32字节查分绝对值求和指令（MPSADBW），其完整函数定义为\_\_m128i \_\_mm\_mpsadbw\_epu8(\_\_m128i a,\_\_m128i b, const int mask)。该指令功能很强大，能同时执行8个4字节的SAD（Sums of Absolute Differences）运算。并且该指令很灵活，能通过选取不同的mask值来改变源操作数和目的操作数的位置。该掩码值虽然是32位数值，但指令只用后后三位。其中后两位用来选取源操作数的连续4个字节的起始位置。掩码的第三位用来选取目的操作时连续11个字节的起始位置。

该指令的运算过程类似于如下代码，如图3.3所示。

i = mask2 \* 4

j = mask0-1 \* 4

for (k = 0; k < 8; k = k + 1) {

t0 = abs(a[i + k + 0] - b[j + 0])

t1 = abs(a[i + k + 1] - b[j + 1])

t2 = abs(a[i + k + 2] - b[j + 2])

t3 = abs(a[i + k + 3] - b[j + 3])

r[k] = t0 + t1 + t2 + t3

}

# 7.字长翻转指令

wsblend指令用于混合两个ω位的参数。

如图所示，即是将a串后一半与b串的前一半链接。

实现该指令的三条指令如右所示。

第一条指令\_\_mm\_blend\_epi16的作用是根据参数c将a和b这两个128位的数值混合在一起，即。

第二条指令\_\_MM\_SHUFFLE是一条宏指令，用来直接在寄存器里调整数值里分量的顺序。

第三条指令\_\_mm\_shuffle\_epi32的作用是根据上一条指令定义的SHUFFLE值跳帧128位的数值h。在这里h被划分成4个int值。例如假如，则得到的。

# 8.字长CRC码计算指令

wscrc指令通过传入一个8bytes长度的字符串来生成一个32bit的CRC值。该指令通过该SSE指令\_\_mm\_crc32\_u64(α)来模拟，由于该指令只有一个时间周期，因此具有很快的速度和很好的鲁棒性。

但因为该指令生成的是64bit的数字，算法中的查询数组大小并不会设置2的64次方这么大，所以需要将其与上（2的n次方-1），即取crc码的后n位值。

# 9.iepsm-1

最早在设计算法时，是打算根据QS算法加上块字符进行改进。因为QS算法的特点在于每次通过读取扫描窗口的下一个字符来查询下一次需要跳转的距离。根据这一特性，可知qs算法最适用于大字典的情况，因为若字典较小，如字典大小为4的基因数据，其跳转长度很有可能分布1-4。因此最初的想法是利用块字符将一组字符看做整体，这样就增大了字典大小。最后这一算法也获得了较好的效果。但在编程调试中偶然发现若只使用定长跳转，就是指针p一直+=常量，能比使用跳转数组获得更好的性能，并且即便这个常量没有跳转数组里面的值大，当然也不能差的太远。可能是现代CPU指令预测特性。所以基于这种现象，我重新去改善算法，并且算法的关键点在于找出一个安全的常值跳转长度，使其不发生漏扫，同时也能跳跃距离最大。

为实现这一点，我借鉴了KR算法的一些特点，因为KR算法是通过算模式串的hash值，再去判断是否需要调用全匹配函数进行一一匹配，但该算法的缺点在于跳转距离过于保守，只有1。

先介绍一下初始化阶段，定义一个指针数组，数组中每项是一个链表，如Node所示，其中next指针指向下一个结点，pos代表定长的子串在模式串中的位置，在初始化阶段对定长的子串调用CRC计算指令得到其hash值，然后将该子串的位置存入对应的数组项中。若相同的子串在模式串中的多个位置出现，即有hash冲突时，则将其存入数组项的下一个链表结点中。

有了这个初始化数组，接下来我们看看如何找出合适的跳转常值。如图所示。

有一串文本，其中模式串长度为5，采样串长度为2，首先将指针指向5-2，跳转值设为5-2+1。也就是说指针每次向前移动四个，然后读取2个字符的采样串，算出hash值，根据初始化数据判断该子串是否在模式串中，不在，继续向前移动四个字符；若在其中，假如是2，表明该子串在模式串的第二个字符起的位置，再调用全匹配，若链表有多个结点，则需继续遍历结点判断。同时也可看出，4是不漏扫的最大长度。

也就是说，加入采样串长度为a，模式串长度为m，则跳转长度为m-a+1。由于生物序列的表现与随机串类似，所以极少出现类似AAAAAAAAAAA这种序列的情况。

所以该算法在实际表现中，文本串的指针基本会一直按常值往前走。

另外对于采样串长度的选择需要通过一组实验去获得。因为a越小，跳转长度越长，但匹配概率会增大，即会增加调用memcmp函数概率；同时a对跳转长度的影响也会随着串长的增加而减弱，因为a的变化区间只在4到8之间，而串长则从12到2048甚至更长。

# 10.iepsm-2

该算法只处理串长大于等于4字节且小于12字节的情况。对于这种短串情形，本来也可以根据长串处理的思想去采样4字节算hash值，因为SSE指令一次能处理128位的数据，所以我们可以将4字节的采样串一次性处理，减少读表的步骤。即通过字长匹配指令读取头四个字符，该指令能直接返回这四个字符在文本串中的匹配情况，若结果中的某一位指示匹配上了，再调用全匹配函数去判断后几个字符是否匹配。

# 11.算法实验

这是针对长串算法中参数调优的实验，选取的数据为人类的基因数据和蛋白质数据，每项的最优值为加黑的斜体字。

我们发现，对大部分数据而言，存在表4.8和表4.9中所列的优化参数。对于不同类型的测试数据（即不同的字典大小）来说，采取如下的块字符大小能获得最优性能。

接下来是算法对比实验，该实验中iepsm使用了上个实验获得的最优参数。

实验选取的测试数据集涵盖了三种不同类别的生物，并且串长度也从1.4MB到3GB之间变化。具有一定代表性。

比对算法选了5个，有针对生物序列的tvsbs，根据传统分类方法的三种类别下的最新串匹配算法，还有使用SIMD指令运用块字符的ssecp算法。

实验结果如图所示，易知iepsm较于其他算法有较大优势，只有在长串处理时，ssecp的性能才接近iepsm。并且算法对于不同种类和不同长度的数据都保持了很好的稳定性。

# 12.云平台集成

由于基因所也正在搭建Hadoop平台，所以根据生物信息学的实际需求，并考虑到生物序列串的海量存储规模，因此采用Hadoop平台去集成这一算法。

即首先用户输入模式串和需要查询的基因类别（不填则全库查询），由于输入串是fasta格式的数据，需要对其去除注释信息。然后根据类别定位到对应的FTP地址（这里主要是美国国立生物技术信息中心的数据ftp://ftp.ncbi.nlm.nih.gov），先判断是否需要解压后，再根据基因文件大小来判断使用单节点下载数据去处理，还是分成小任务由各个节点分别定位到不同的位置去获取。每个任务在获得文件后，根据其对应的文件格式选取对应的优化参数，再调用iepsm算法去统计匹配个数和位置，最后合并各个子任务的结果并在web页面上展示。

系统的输入界面如图5.2所示，左图的input输入框内可填写自定义查询基因串，右图的upload框内可选择上传fasta的格式基因片段文件。

系统的结果显示界面如图5.3所示，匹配的位置都被标注出了。