文章编号：1003-0077（2017）00-0000-00

**基于指令级并行的快速求交算法**

宋省身 杨岳湘

（国防科学技术大学 计算机学院，湖南省 长沙市 410000）

**摘要：**布尔查询中的求交操作被广泛应用于各种信息系统中，是进行文档检索的基本操作之一。其基本形式可以视作多个有序整数序列的交集问题。为了提高求交运算的效率相继有许多算法被提出，近几年学者开始使用并行指令集（SIMD）来加速传统的求交算法，并取得了良好的效果。然而，这些优化仅局限于使用在两两求交的场景中，传统的多序列求交算法难以与之匹配。本文首先回顾了传统的多序列求交算法，然后针对其核心的搜索算法提出了两种基于SIMD的优化方式。实验验证，优化后的算法相比未使用SIMD的情况下有了很大的提升，甚至优于SIMD优化后的两两相交算法

**关键词：**求交算法；倒排索引；性能评价；并行处理

**中图分类号：**TP391  **文献标识码：**A

#### Efficient Multiple Sets Intersection using SIMD Instructions

Xingshen Song, and Yuexiang Yang (College of Computer, National University of Defense Technology, Changsha, Hunan 410000 ,China)

**Abstract :** Conjunctive Boolean query is one fundamental operation for document retrieval in many information systems and databases. In its most basic and popular form, a conjunctive query can be seen as the intersection problem of multiple sets of sorted integers. Various algorithms have been put up in terms of maximizing the query efficiency. In recent years, researcher began to exploit the parallel advantage of single-instruction-multiple-data (SIMD) instructions to accelerate the intersection procedure and achieved substantial gains over previous scalar algorithms. However, these works only focus on intersecting two sets at a time rather than the whole lot of them.

Missing from the literature is a thorough study that explores the combination of traditional multiple sets intersection algorithms and SIMD instructions. In this article, we first revise all the traditional algorithms, then propose two optimizations on the essential searching algorithms using SIMD instructions. Experiments show that the optimized algorithms performs much better than the traditional ones, even outperform the recent SIMD intersection algorithms.

**Key words:** Set Intersection, Performance Evaluation, Vectorized Processing

**1 引言**

布尔查询中的合取操作是信息检索中最常用的查询方式之一，其他复杂的查询方式如排序查询，模糊查询都建立在它的基础之上。合取操作，也就是对多个集合中的元素求取交集，常见于对文档内容的检索中。在当前的信息检索系统中，倒排索引作为一种简单高效的数据结构，被广泛应用于网页数据的存储和查询中。倒排索引的结构由两部分组成，其中，词典用于存储查询词，并将其映射到相应的倒排链表中，倒排链表则按序存储了查询词出现的文档号、频率和位置信息。通常，这些信息都是以整型数的格式进行操作，因此，处理查询词的过程也可以视作对其相应的个倒排链中的整型数求取交集的过程。当前的互联网数据和用户持续增长，为查询处理的效率带来了巨大的压力，为了解决这些问题，相继有许多算法被提出以提高求交操作的效率，最早的工作可以追溯到几十年前。

尽管算法众多，但很难有一个通用的方法能够适用于各种场景下的求交操作，参与的倒排链的数量，倒排链的长度比以及相交元素的数量分布都会影响到算法的效率。除此以外，不同的算法也拥有各自的特点和适用的情况。最直接的一种求交方法叫做Zipper，通过对倒排链两两逐项求交的方式进行计算。另一个针对两两求交的算法叫做BaezaYates，它采用一种分治的方法不断地二分查找出相交元素，其工作过程类似于快速排序。当倒排链表的数量超过两个时，另一类算法不再采用两两相交，而是把所有参与的倒排链视作一个整体同时求交，这类算法每次从某一倒排链中选出一个元素作为*eliminator*，之后在其他链表中进行查找。根据选定和查找方式的不同，这些算法可分为set\_vs\_set，Adaptive，Sequential和MaxSuccessor。

除此以外，还有方法从不同的角度来解决这个问题。某些方法使用了辅助性的数据结构如树堆（treap）、小波树（wavelet tree）、跳表（skip list）和哈希表（hash table）等实现了对链表内任意元素的快速访问，并以此减少了许多不必要的比较；某些方法利用了当前硬件的特性如GPU和多核技术来实现并行计算；还有一些方法研究了如何使用哈希结构快速估计交集大小而不是精确计算交集的问题。然而这些方法或者需要额外的预处理和存储空间，或者对输入倒排链存在特殊的要求。

近年来，研究开始使用并行指令集来加速倒排链求交操作的过程，通过使用代码级的并行指令和更大的寄存器空间，基于单指令多数据流（single instruction multiple data, SIMD）的方法能够一次比较4个甚至更多的整型元素，从而极大地提升传统算法的速度。相比于使用GPU和多核技术的优化算法，SIMD提供了更加简单易用的接口和灵活高效的实现方式，在倒排索引的压缩算法中已经得到了广泛的应用。但在查询处理，尤其是布尔查询中，SIMD的应用还存在一些提升空间。当前的优化方法仅仅针对倒排链两两相交的情况，工作方式更类似于并行化的Zipper，使其适用范围局限在了长度相近并且交集较大的倒排链求交情景中。而传统的面向多个倒排链求交的算法并没有得到关注，并行指令集和多倒排链求交算法的结合在研究领域内尚属空白。为此，本文首先回顾分析了传统多倒排链表求交运算中具有代表性的方法以及其基于SIMD的优化，然后针对多倒排链求交算法提出了两种并行化的搜索算法，实验证明，

**2 相关工作**

倒排链求交问题一直是信息检索领域中的热门研究点，各种各样的算法和数据结构不断地被提出以提高这一问题的计算效率。除去某些特殊的研究点（如快速估计交集大小上限），本文仅考虑精确计算出交集的求交算法。同时，压缩状态下的倒排链或者增加了辅助数据结构的求交算法同样不在本文的研究范围，因为它们在便利了某一种计算场景的同时，也会限制算法在其他场景下的应用。本文所使用的倒排链均视作在压缩展开后的状态下。

**2. 1 多倒排链求交算法**

传统的串行求交算法主要关注如何减少不必要的比较。针对查询词，对应的倒排链为，其长度，算法的基本流程主要包含两个部分，一是从中选取*eliminator* ，二是从其他倒排链中使用特定的搜索算法*F-Search*找到不小于的元素。如果在所有倒排链中都被找到，那么称之为一次*full match*，否则是*mismatch*，无论如何，都会被更新，然后开始新一轮的搜索直至任一倒排链遍历完毕。按照处理倒排链的方式，可将这些算法分为两类，一类为两两相交，一类为整体相交。

其中，两两相交的通过每次选取两条最短的倒排链求取交集，再把结果与剩余倒排链中最短的重新求交，直至全部遍历完毕。Zipper和BaezaYates是两种典型算法。Zipper使用了线性查找（Linear search）作为*F-Search*，其时间复杂度为相交两倒排链长度之和，即。而BaezaYates则是基于二分查找（Binary search），不断地在长倒排链中寻找短倒排链的中值，如果找到则加入交集中；随后两个倒排链均被拆分为两部分，查找继续递归进行，直至任一方变为空集，其平均时间复杂度为。

整体相交方法同时搜索所有的倒排链，类似于排序查询中的Document-At-A-Time（DAAT）策略，其伪代码如图1所示。其中，最简单的策略叫做set\_vs\_set（svs），每次都是从中选取，并在其他倒排链中二分查找。然而svs没能利用倒排链中的数值分布特性，查找过程中其他倒排链很有可能取代成为最短倒排链，为此Dmaine等人提出了Swapping\_svs，在每次更新后，按照剩余倒排链长度进行重排序。Adaptive和Sequential是应用较为广泛的算法，它们使用了一种新的*F-Search*，叫做Galloping search。即从初始步长1开始，不断倍增步长，直至找到不小于的元素位置，然后在当前步长内二分查找到具体位置。而如果发生了*full match*，从当前链的下一个位置选取，否则由引起*mismatch*的元素替代。Sequential和Adaptive唯一的区别，就是在执行Galloping search时，Adaptive每次循环只向前移动一次，而Sequential则是直至发现不小于的值才停止。尽管二者的时间复杂度相同，实验显示Adaptive在相交元素比较密集时表现出色，而后者更适用于交集较大的情况。结合Swapping\_svs的特点，Demaine等人又提了Small Adaptive，在每次更细之前按照长度对倒排链进行重排序，以保证每次都从最短的倒排链中选取。Clupepper和Moffat认为重排序会引入额外的时耗，为此他们提出了Maxsuccessor（max）来克服这一问题，即维持原来的访问顺序，而由的下一个元素和引起*mismatch*二者中的大值决定。以避免选择了无效的后引起的冗余比较。

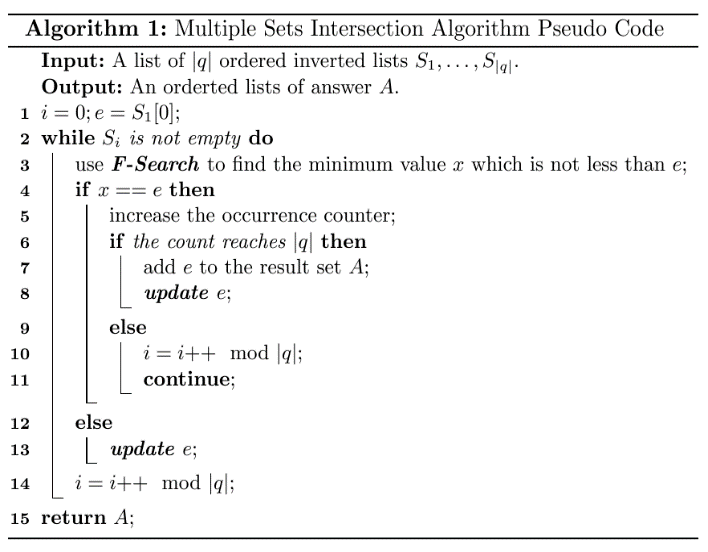


图1 多倒排链求交算法的伪代码

从图1的算法中，可以看到*F-Search*是算法中每一次时耗最长的部分，为此出现了很多搜索算法，除了之前提到的Linear search, Linary search和Galloping search，还有Golomb search, Fibonacci search以及Interpolation search。Golomb search同样使用跳查，但是用的是固定步长不是Galloping search的倍增过程；Fibonacci search与binary search类似，不同于二分，前者选择的是按照两个相邻的Fibonacci数进行分治；Interpolation search则是通过线性回归的方式缩小相交位置可能出现的范围。

**2. 2 基于SIMD的求交算法**

近年来，研究者开始使用并行指令集来加速信息检索领的传统算法。通过使用长度为128bit的寄存器（成为XMM寄存器），指令可以同时操作4个32bit的整型数，近年来还出现了256bit的YMM，甚至512bit的ZMM。目前主要针对索引压缩和求交两个方面，而并行指令的引入使效率得到了极大地提升。

Schelegel等人使用了Intel SSE4.2中的STTNI指令集合，能够在一次调用中比较16个8-bit或者8个16-bit的整型数，这样在两个倒排链之间既可以以块为单位进行比较。对于长度超过16bit的数字，他们设计了一种层次化结构来分开存储高低16bit部分，只有两部分完全相同时，才记为一次相交。该算法对两个长度相当并且值域较小的倒排链比较有效，但指令消耗的时钟周期相比于一般的SIMD指令要更高，因此提升效果有限。Lemire选择每次从短倒排链中取出一个元素，然后同时与长倒排链的多个元素进行比较，这种算法可以视作并行化地Golomb search，长倒排链的指针每次移动一个或者多个分块的长度，直至找到可能相交的分块，并在分块内使用二分查找，为了适应不同长度的倒排链，他们提出了V1，V3和SIMD GALLOPING三种算法。Inoue等人则是针对减少难以预测的*if\_greater*判断，重新组织了存放在寄存器中的元素，通过使用代价更小的*if\_equal*判断仅比较头部字节的情况下，排除掉大多数不相交元素。

上述三个算法尽管在效率上都取得了显著的提升，但仅考虑了两两相交的情况，并没有针对多倒排链相交做过验证，而传统多倒排链相交算法的优势并没有得到利用，这就制约了它们的适用范围。为此，我们介绍了两种并行优化的多倒排链求交算法。

**3 并行的多倒排链求交算法**

从图1中的算法可知，选择和*F-Search*是影响算法效率的重要因素，

**1. 1 标题** 二级标题五号黑体

正文

1.2.1 标题三级标题五号宋体

正文

**2 标题**

**参考文献**

参考文献小五号，只列举最主要的，必须是公开发表的书刊才能列入，最少不得少于5条。文献按文章中出现先后顺序排列

**(各类文献严格按照主页上的《参考文献规范》)**

1. Culpepper J S, Moffat A. Efficient set intersection for inverted indexing[J]. Acm Transactions on Information Systems, 2010, 29(1):1.
2. Zobel J, Moffat A. Inverted files for text search engines[J]. Acm Computing Surveys, 2006, 38(2):6.
3. 闫宏飞, 张旭东, 单栋栋,等. 基于指令级并行的倒排索引压缩算法[J]. 计算机研究与发展, 2015, 52(5):995-1004.
4. 论文集：［序号］ 作者.题名[C]//编者.论文集名.出版地：出版者，出版年，起止页码.
5. 学位论文：［序号］ 作者.题名[D].保存地点：保存单位，年份.
6. 报告：［序号］作者.题名[R].保存地点：保存单位，年份.
7. 报纸文章：［序号］ 作者.题名[N].报纸名，出版日期(版次).
8. 标准：［序号］标准编号，标准名称[S].
9. 专利: ［序号］专利所有者.专利题名[P].专利国别:专利号,公开日期.
10. 电子文献:主要责任者.电子文献题名[电子文献标识/载体类型]. [发表或更新日期].电子文献的出处或可获得地址.
11. 电子文献标识:[DB]-数据库 [CP]-计算机程序 [EB]-电子公告
12. 电子文献载体类型:[OL]-联机网络 [MT]--磁带 [DK]-磁盘 [CD]-光盘