# 多核的并行相似连接

冯林静
(天津工业大学,天津 300387)

**摘 要**:相似连接(similarity join)是指在给定的数据集中,根据给定的相似度度量函数来衡量数据之间的相似度,并找出所有相似度不小于给定阈值的数据对的操作。随着网络和移动应用等信息技术的不断发展,数据呈现爆炸式增长,海量数据的分析需要强大的计算能力,相似连接成为大数据处理领域的热点方式之一。传统的单核计算机平台的处理能力已经很难满足海量数据处理的计算要求。为了提高计算效率和性能,利用基于多核平台的多线程并行编程发挥多核体系结构的优势,已经成为实现个人低成本并行计算和多核技术发展的趋势。因此,为了提高相似连接的效率,充分利用现代体系结构的多核特性和多线程技术,提出了相似连接并行化的改进方法。实验结果表明,使用该方法极大地提升了效率。

**关键词**:多核;多线程;并行;相似连接
**中图分类号**:TP311 **文献标识码**:A **文章编号**:1673-629X(2017)07-0043-04
doi:10.3969/j.issn.1673-629X.2017.07.010

# Parallel Similarity Join of Multi-core

FENG Lin-jing
(Tianjin Polytechnic University, Tianjin 300387, China)

**Abstract**: Similar join is an operation which is using a given similarity function to measure the similarity between data and find out all similarity less than a given threshold in a given data set. With the continuous development of Internet and mobile applications, the amount of data is increasing explosively, and along with the analyzing of huge amount of data, it requires a strong ability of calculation, so similar joins become one of the leading way of hotspots in the field of data processing. The processing capacity of traditional single-core computer platform has been difficult to meet the calculation of mass data processing requirements. Programming based on multi-core platform and using the multi-thread parallel can make full use of the advantage of multi-core architecture and improve the computational efficiency and computational performance, which has become the trend to realize personal low cost calculation and the development of multi-core technology. Therefore, based on the characteristics of multi-core and multi-thread technology, the improved method of similar connected parallelization is proposed. The experimental results show that the efficiency has been obviously improved.

**Key words**: multi-core; multi-thread; parallel; similar join

## 0 引 言

在大数据背景下,数据加速增长,如何从中筛选出有用的数据,高效地进行相似连接,显得日益重要[1-3]。相似连接应用广泛,比如文档聚类,找出重复网页集合,检测剽窃,数据集成[4]。已知两个数据集,相似连接是从这两个数据集中找出所有相似的数据对。通常,判定两个数据对是否相似,一个相似函数和一个阈值是必要的。广泛应用的相似函数有:Jaccard similarity、Cosine similarity、Overlap similarity、Hamming distance、Edit distance。文中分别用 Jaccard similarity 和 Hamming distance,以及 Cosine similarity 和 Hamming distance 相结合的方法去量化数据对相似值。

在数据对的相似连接方面有很多研究,大致分为三类:Gram-based method[5-6]、Trie-based method[7-9]、Partition-based method[10]。对这三种方法进行了比较,结果发现,三种方法各有所长。其中,Partition-based method 对短字符串和长字符串的数据集都有很好的性能,Trie-based method 和 Gram-based method 分别对短字符串的数据集和长字符串的数据集有很好的性能。为此,采用 Gram-based method 实现相似连接的框架,把相似连接并行化在多核平台上实现。

**收稿日期**:2016-08-23 **修回日期**:2016-11-24 **网络出版时间**:2017-06-05
**基金项目**:国家自然科学基金资助项目(61402329)
**作者简介**:冯林静(1991-),女,硕士研究生,研究方向为基于多核相似连接。
**网络出版地址**:http://kns.cnki.net/kcms/detail/61.1450.TP.20170605.1509.066.html
万方数据

## 1 多核体系结构

随着时代的发展,单核处理器已经退出了历史舞台。相对于多核处理器,单核处理器具有许多性能瓶颈[11],其局限性表现为:主频低,系统能耗问题突出,对大型功能需求处理能力低等。于是提出了多处理器和多核的概念。其中,多处理器是每个处理器位于一个独立的芯片上,而且有着自己的硬件。多核处理器,也就是多核 CPU,是将两个或两个以上处理器集成在一个芯片上,每个 CPU 都是一个单独的处理器,通过并行总线将各 CPU 连接起来。多处理器和多核处理器结构如图 1 和图 2 所示。

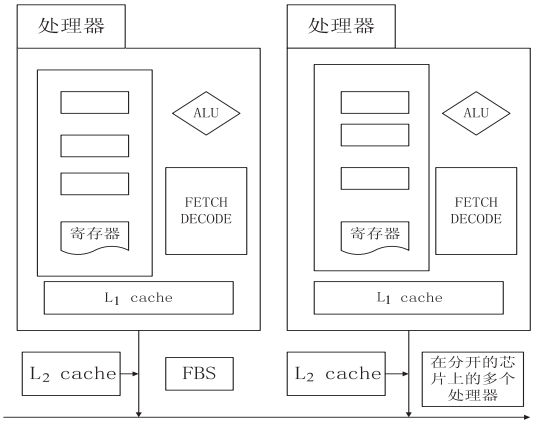


图 1 多处理器

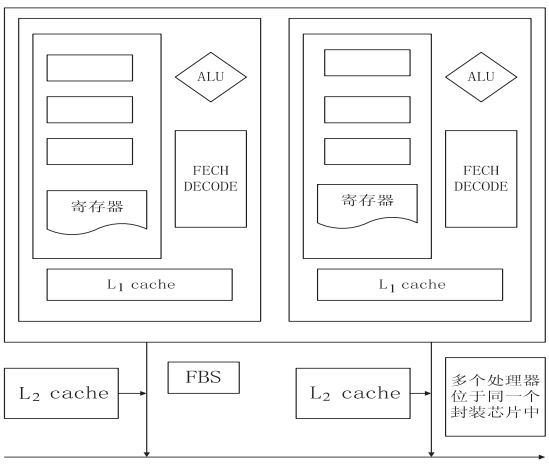


图 2 多核处理器

图 1 中,每个处理器都有较为独立的电路支持,有自己的 cache,处理器之间通过板上总线进行通信。图 2 中,多核处理器可以把任务分配给多个核执行,每个核任务不同,提高了处理速度。而且多核共享内存的特性,相对于多处理器减少了内存延迟和系统调度的消耗。

## 2 问题描述

本节描述在相似函数的约束条件下相似连接生成的数据对[12]。

定义相似连接:从一个有限域 $u = \{w_1, w_2, \cdots, w_{|u|}\}$ 中,定义一组符号作为一条记录。通过相似性函数 sim( ),得到一个相似度值在[0,1]的两条记录。已知记录的集合,一个相似性函数 sim( )和一个相似度阈值 $t$,相似性连接问题是求得所有相似记录对 $\langle x, y\rangle$,也就是两组记录相似值不小于阈值 $t$,即 $\mathrm{sim}(x, y) \geqslant t$。

相似连接函数为:

Jaccard similarity:

$$J(x,y) = \frac{|x \cap y|}{|x \cup y|} \tag{1}$$

Cosine similarity:

$$C(x,y) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|} = \frac{\sum_i x_i y_i}{\sqrt{|x|} \cdot \sqrt{|y|}}$$

Overlap similarity:

$$O(x,y) = |x \cap y|$$

文中使用规范化后的数据集,比如:

$D_x$ = "yes as soon as possible"

$D_y$ = "as soon as possible please"

按照字典排序得到如下有序列表,从而使数据集规范化。

$L_x$ = "as, $as_1$, possible, soon, yes"

$L_y$ = "as, $as_1$, possible, please, soon"

把位置过滤的方法分别用在前缀过滤和后缀过滤上,进一步减小生成相似数据对的数量[13],如算法 1 和算法 2 所示。

```
Algorithm1:PPJoin_Index( D,θ,N )
input: D -collection of data; θ -the similarity threshold; N -the number of threads
output:All pairs of strings <x,y>,if sim( x,y ) ≥ θ
1  //Data Partition
2  for n =0→ N do
3  In←Data Partition( n );
4  //Build Index
5  for each d in In do {
6  ←get prefix tokens of d ;
7  for each t ∈ do
8  . add( t );}
9  //Task Execution
10  for n =0→ N do
11  for each t in U do {
12 . range←{△1( ) ∪ ∪ △2( )};
13  pthread_create( &Id[ n ],NULL,&JoinThread1,&n );
14  }
Algorithm 2:JoinThread1( n )
1  for each d in In do
2  for each t in do {
3  tmp=ProbeIndex( . range);
```

万方数据

```
4  Candidates. add(tmp);}
5  Verification();
6  OutputResults();
```

以 Jaccard similarity 为例求得集合间的相似度。实验部分分别测试了 Jaccard similarity 和 Cosine similarity。

例如:已知 $sim(x,y) = J(x,y)$,$t = 0.8$,从规范化后的数据集中临近选六条数据,如表 1 所示[14]。

计算出前缀长度,建立倒排索引后对数据进行前缀过滤,得到如下相似数据对:

<u,v>,<v,w>,<v,y>,<w,x><w,y>,<w,z>,<x,y>,<x,z>,<y,z>

把位置过滤应用到前缀中过滤:

<u,v>,<w,y>,<w,z>,<x,z>,<y,z>

把位置过滤应用到后缀中过滤:

<u,v>,<x,z>,<y,z>

Max-depth 加 1,递归上一步:

<u,v>

表 1　规范化后的样本数据

| ID | String |
|---|---|
| u | CDEF |
| v | BCDEF |
| w | ABCDF |
| x | GABEF |
| y | ABDEF |
| z | GACDEF |

因此 PPJoin+算法在数据相似连接上效率很高,面对大量数据,还要通过并行化对其进一步优化。

## 3　基于多核的并行相似连接

传统的相似连接算法是在单核平台上实现的,所有数据在同一顺序编程条件下运行,该方法已经无法通过增加时钟频率或处理器速度来提高计算性能。在各种类似需求的情况下,单核已经被淘汰,多核处理器成为发展的潮流[15]。多核处理器的低成本和广泛可用性,使得并行处理不再是超级计算机或者集群的专属领域。可以根据需要利用多处理和多线程的应用。文中在多核处理器上利用多线程的并行化,对相似连接算法进行优化,提高相似连接的速度[16-17]。

多处理器的发展,它的核心问题在软件上。文中是在多核的平台上对相似性连接进行并行化,采用的是多线程的方法。多核环境下软件开发的核心是多线程开发[18],需要处理好的问题有:核内资源竞争;中断处理;处理器间的通信;线程与进程的调度;Cache 一致性。其中,线程的调度和分配是关键点,要根据数据共享情况,分配到物理核上。

具体工作中,并行化编程中非常重要的步骤就是工作分解。分解是对基础算法进行分析,将之分解为若干相对独立的部分或者操作,进而可以把这些相对独立的部分分配到多个处理单元执行。工作分解一般可以分为任务分解和数据分解。任务分解是把一个算法按照操作的相关性分解为若干可以同时执行的子任务。数据分解则是将一个比较大的待处理的数据集分割为若干子集,从而可以使不同部分的成员实施同时的运算或操作。文中把 suffix-filtering 部分进行任务分解,如果开 $n$ 个线程,把数据集分成 $n$ 份,数据分别在不同的线程上运行。

在串行程序设计过程中,为了节约带宽或者存储空间来提高程序的性能,比较直接的方法,是针对数据结构做一些有目的的设计,将数据压缩得更紧凑,减少数据移动。但是在多核多线程体系中,这种方法有时会起到相反的作用。数据不仅在存储区和执行核之间移动,同时还会在执行核之间进行传输。依据数据的相关性,其中有两种读写模式会与数据的移动相关联:写后读和写后写。由于这两种模式会引发数据之间的竞争,表面上是并行执行,但实际却只能串行执行,进而影响到计算性能。那么,如果将进程和线程与 CPU 绑定,最直观的优势就是提高了 CPU Cache 的命中率,从而减少内存访问损耗,提高程序的运行速度。这也就是 CPU 亲和(affinity),让进程在某个给定的 CPU 上尽量长时间地运行而不被迁移到其他处理器的倾向性。Linux 内核进程调度器本身就具有被称为 CPU 软亲和性的特性,这意味着进程通常不会在处理器之间频繁迁移。利用这个优势,进程迁移的频率就会降低,也就意味着产生的负载小,但不代表不会进行小范围的迁移。

另外,CPU 硬亲和性是让进程锁定在某个处理器上运行,而不是在不同的处理器之间进行频繁的迁移。利用这些特性不仅可以改善程序的性能,同时还可以提高程序的可靠性。

文中选用提供了完备 API 的 Linux Pthread 多线程库。Pthread 更底层,并且提供了虚拟编写任何可知线程行为的能力,非常灵活,每个线程行为的每个细节都可以自行定义。

多线程的使用在提高性能的同时也带来了一些新的问题,主要是如何实现导致两个线程同时对一个对象进行访问或者修改,这就有可能导致程序错误。例如:如果一个线程要对一个对象进行读写操作,同时另一个线程也要对该变量进行读写操作,这样前一个线程读入的变量值可能就是一个不稳定的值。因此,在多线程编程中,要采取互斥和条件同步的方法来同步线程的执行。

万方数据

## 4 实 验

### 4.1 实验配置

实验平台配置:Linux CentOS release 6.2(Final),3 GB 主存,程序用 C++编写,编译器为 GCC 4.4.7。实验中使用的数据集为 DBLP。DBLP 是一个文献出版信息的数据集,其中包含 1 021 062 条记录。每一条记录中提取出文献名称和作者信息作为一条新的记录并以字符串的形式保存。测试比较了相同的数据集在多核平台下,分别用 Jaccard similarity 函数和 Cosine similarity 函数所运行的时间。

### 4.2 实验结果分析

选用一百万条数据进行实验。用 Jaccard similarity 和 Cosine similarity,把在单核平台和在多核平台下实验的结果进行比较,把在多核平台下开启不同数量的线程进行比较。在利用多线程方法设计程序时,如果产生的额外消耗大于线程本身的工作任务,就不必采用并行了。开启的线程数量并不是越多越好,软件线程的数量尽可能与硬件线程的数量相匹配。最好根据实际需求,通过不断的实验和调优,来确定开启线程数量的最佳值。

用 Jaccard similarity 和 Cosine similarity 计算,其中列 1 是在单核平台下的结果,列 2、3、4 分别是多核平台下开启不同个数线程运行的结果。

串行程序中,在相同的 Threshold 条件下,用相同的数据集 Jaccard similarity(系列 1)和 Cosine similarity(系列 2)进行过滤,相似连接所消耗的时间如图 3 所示。

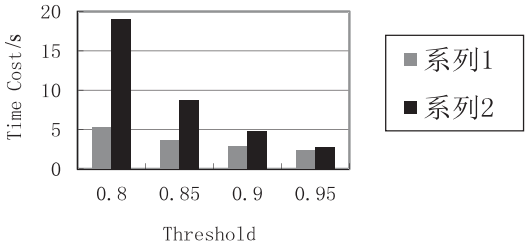


图 3 不同约束条件的比较

在并行程序中,开启 2 个(系列 2)、3 个(系列 3)、6 个(系列 4)线程时,用 Jaccard similarity 和 Cosine similarity 约束条件所消耗时间和原来串行程序(系列 1)下所消耗时间进行对比,分别如表 2 和表 3 所示。

表 2 Jaccard similarity 约束条件下的比较

| 阈值 | 开启线程个数 | | | |
|---|---|---|---|---|
| | 0 个 | 2 个 | 3 个 | 6 个 |
| 0.8 | 5.329 | 3.96 | 4.125 | 5.418 |
| 0.85 | 3.645 | 2.724 | 3.092 | 3.991 |
| 0.9 | 2.858 | 2.268 | 2.53 | 3.452 |
| 0.95 | 2.249 | 2.046 | 2.244 | 3.018 |

万方数据

表 3 Cosine similarity 约束条件下的比较

| 阈值 | 开启线程个数 | | | |
|---|---|---|---|---|
| | 0 个 | 2 个 | 3 个 | 6 个 |
| 0.8 | 18.966 | 13.483 | 14.568 | 15.528 |
| 0.85 | 8.669 | 6.619 | 7.088 | 7.228 |
| 0.9 | 4.72 | 3.437 | 3.805 | 4.799 |
| 0.95 | 2.746 | 2.198 | 2.455 | 3.378 |

由此可知,多核平台下数据运行效率有所提高。

## 5 结束语

提出了一种基于多核的并行相似连接的方法。利用相似连接必要条件,选用 Jaccard similarity 和 Cosine similarity 作为约束条件进行过滤。为了提高效率,利用多核平台,对算法进行并行化。实验结果表明,该方法能在不同的相似度约束、不同的数据集上更高效地实现相似连接。由于现代的处理器都采用了多级 cache,在文中的研究过程中发现多级 cache 对并行算法的性能影响很大,因此如何更好地利用多级 cache 是下一步的研究工作。

**参考文献:**

[1] 刘雪莉,王宏志,李建中,等. 基于实体的相似性连接算法[J]. 软件学报,2015,26(6):1421-1437.

[2] Li G, Deng D, Wang J, et al. Pass-join: a partition-based method for similarity joins[J]. Proceedings of the VLDB Endowment,2011,5(3):253-264.

[3] Xiao C, Wang W, Lin X, et al. Top-k set similarity joins [C]//25th international conference on data engineering. [s. l.]:IEEE,2009:916-927.

[4] Xiao C, Wang W, Lin X, et al. Efficient similarity joins for near duplicate detection[J]. ACM Transactions on Database Systems,2008,36(3):563-574.

[5] Li C, Lu J, Lu Y. Efficient merging and filtering algorithms for approximate string searches[C]//24th international conference on data engineering. [s. l.]:IEEE,2008:257-266.

[6] Xiao C, Wang W, Lin X. Ed-Join: an efficient algorithm for similarity joins with edit distance constraints[J]. Proceedings of the VLDB Endowment,2008,1:933-944.

[7] Ji S, Li G, Li C, et al. Efficient interactive fuzzy keyword search[C]//International conference on world wide web. [s. l.]:[s. n.],2009:371-380.

[8] Li G, Ji S, Li C, et al. Efficient fuzzy full-text type-ahead search[J]. The VLDB Journal,2011,20(4):617-640.

[9] Wang J, Feng J, Li G. Trie-join: efficient trie-based string similarity joins with edit-distance constraints[J]. Proceedings of the VLDB Endowment,2010,3(1-2):1219-1230.

[10] Wang J, Li G, Fe J. Fast-join: an efficient method for fuzzy to-

(下转第 50 页)

实验结果表明,使用提出的方法仅仅只是检索出一个特征相关的源代码,计算查全率和调和平均数就没有意义,所以就不进行定义。这样的评价方法本质上是将检索量从实验数据总数的 10% ~15% 缩减到 1 个,但却大大压缩了特征定位的成本,提高了软件演化的效率。

## 4 结束语

特征定位是软件过程及软件演化等活动得以顺利展开的保证。特征定位研究结果的质量影响着软件演化活动的效率,而特征定位研究结果的评估标准决定了软件演化活动的波及范围。结合波及效应分析及当前常用的评价标准,提出了一种新的实验评价方法,所提出的评估标准可以大大降低软件演化活动的工程量及工程范围,所采用的评价标准不但具有普遍意义,更具有实际意义。

**参考文献:**


[1] Dit B,Revelle M,Gethers M,et al. Feature location in source code:a taxonomy and survey[J]. Journal of Software Maintenance & Evolution Research & Practice,2013,25(1):53-95.

[2] Abebe S L,Alicante A,Corazza A,et al. Supporting concept location through identifier parsing and ontology extraction [J]. Journal of Systems and Software,2013,86(11):2919-2938.

[3] Scanniello G,Marcus A,Pascale D. Link analysis algorithms for static concept location:an empirical assessment[J]. Empirical Software Engineering,2015,20(6):1666-1720.

[4] Wilde N,Gomez J,Gust T,et al. Locating user functionality in old code[C]//Proceedings of international conference on software engineering. [s. l. ]:IEEE,1992:200-205.

[5] Alhindawi N,Alsakran J,Rodan A,et al. A survey of concepts location enhancement for program comprehension and maintenance[J]. Journal of Software Engineering and Applications,2014,7(5):413-421.

[6] Dit B,Revelle M,Poshyvanyk D. Integrating information retrieval,execution and link analysis algorithms to improve feature location in software[J]. Empirical Software Engineering,2013,18(2):277-309.

[7] Poshyvanyk D,Gueheneuc Y G,Marcus A,et al. Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval[J]. IEEE Transactions on Software Engineering,2007,33(6):420-432.

[8] Bohner S A. Impact analysis in the software change process:a year 2000 perspective[C]//International conference on software maintenance. [s. l. ]:[s. n. ],1996:42-51.

[9] Li B,Sun X,Leung H,et al. A survey of code-based change impact analysis techniques[J]. Software Testing Verification & Reliability,2013,23(8):613-646.

[10] 王 炜,李 彤,何 云,等. 一种软件演化活动波及效应混合分析方法[J]. 计算机研究与发展,2016,53(3):503-516.

[11] Rajlich V,Gosavi P. Incremental change in object-oriented programming[J]. IEEE Software,2004,21(4):62-69.

[12] Li T. An approach to modelling software evolution processes [M]. Berlin:Springer,2009.

[13] Seacord R C,Plakosh D,Lewis G A. Modernizing legacy systems:software technologies,engineering processes,and business practices[M]. [s. l. ]:Addison-Wesley Professional,2003.

[14] 鞠小林,姜淑娟,张艳梅,等. 软件故障定位技术进展[J]. 计算机科学与探索,2012,6(6):481-494.

[15] Marcus A,Sergeyev A,Rajlich V,et al. An information retrieval approach to concept location in source code[C]//11th working conference on reverse engineering. [s. l. ]:IEEE,2004:214-223.

[16] 韩俊明,王 炜,李 彤,等. 演化软件的特征定位方法[J]. 计算机科学与探索,2016,10(9):1201-1210.

[17] 何 云,王 炜,李 彤,等. 面向行为主题的软件特征定位方法[J]. 计算机科学与探索,2014,8(12):1452-1462.


(上接第 46 页)


ken matching based string similarity join[C]//27th international conference on data engineering. [s. l. ]:IEEE,2011:458-469.

[11] 谢子光. 多核处理器核间通信技术研究[D]. 成都:电子科技大学,2009.

[12] 徐媛媛. 基于 MapReduce 的相似性连接研究[D]. 宁波:宁波大学,2014.

[13] 庞 俊,谷 峪,许 嘉,等. 相似性连接查询技术研究进展[J]. 计算机科学与探索,2013,7(1):1-13.

[14] Chaudhuri S,Ganti V,Kaushik R. A primitive operator for similarity joins in data cleaning[C]//Proceedings of the 22nd international conference on data engineering. [s. l. ]:IEEE,2006:5.

[15] 蔡进国,郭 宏,李伟强,等. 多核多线程环境下的程序并行优化方法[J]. 现代计算机,2004(3):3-5.

[16] Jiang Y,Deng D,Wang J,et al. Efficient parallel partition-based algorithms for similarity search and join with edit distance constraints[C]//Proceedings of the joint EDBT/ICDT Workshops. [s. l. ]:[s. n. ],2013:341-348.

[17] 于 方. 多核平台下的多线程并行编程[J]. 阴山学刊:自然科学版,2010,24(3):33-36.

[18] 眭俊华,刘慧娜,王建鑫,等. 多核多线程技术综述[J]. 计算机应用,2013,33(S1):239-242.


万方数据