ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2014/08(00)-0000-00
doi: 10.3778/j.issn.1673-9418.

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-89056056

大数据下利用块依赖的并行实体解析算法

王宁[1+]，黄敏[1]

1. 北京交通大学 计算机与信息技术学院 北京 100044

Parallel Entity Resolution Based on Block Dependency in Big Data

Wang Ning[1+], Huang Min[1]

1. School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China
+ Corresponding author: Phn: +86-136-9307-1879, E-mail: nwang@bjtu.edu.cn

Wang Ning, Huang Min. Parallel Entity Resolution Based on Block Dependency in Big Data. Journal of Frontiers of Computer Science and Technology, 2014, 8(0): 1-000.

Abstract: Entity resolution(ER) is widely used in database management and information retrieval. In the age of big data, entity resolution faces new challenges in dealing with mass data. An algorithm is proposed for parallel entity resolution based on block dependency to adapt to big data environment, which consists of three stages under MapReduce programming framework. Firstly, blocking is helpful for reducing the amount of calculation. Secondly, the entities which are of low dependency to the block that they belong to are picked out to match entities in other blocks. Using this kind of filtering strategy, not only the accuracy of resolution results is kept, but also the amount of calculation is reduced in some degree. Lastly, span distance is set to control the resolution quantity and further improve the efficiency. By evaluating on Hadoop using real data set, experimental result shows that our algorithm is efficient and effective.

Key words: entity resolution; big data; MapReduce; blocking; data screening

摘 要: 实体解析在数据库管理、信息检索中均有广泛应用，大数据时代的到来使得实体解析在海量数据的处理上面临新的挑战。为适应海量数据的实体解析工作，提出基于块依赖的并行实体解析方法，该方法在 MapReduce 编程框架下分三阶段实现：首先，依靠分块技术初步减少计算量；其次，通过基于块依赖的

The National Natural Science Foundation of China under Grant No.61370060,(国家自然科学基金); the Jiangsu Provincial Natural Science Foundation of China under Grant No.BK2011454,(江苏省自然科学基金).
Received 2000-00, Accepted 2000-00.

数据筛选策略，挑选分块中与所在块的块依赖度较低的实体与其他块中实体匹配，不仅保证解析结果的正确性，且在一定程度上控制了计算量；最后，通过设定跨度距离来控制解析数量，进一步提高时间效率。采用真实数据集在 Hadoop 上对该方法进行了评估，实验结果表明该方法在保证解析质量的基础上具有良好的时效性。

关键词: 实体解析；大数据；MapReduce；分块；数据筛选

文献标志码：A **中图分类号：**TP311

1 引言

实体解析[1]是指对现实世界中同一实体的不同表现形式进行识别、连接和分组，它在数据库管理、信息检索、机器学习中都有广泛应用。它可以帮助客户找到不同商家对于同一产品的不同描述和价格定位，也可以为人口普查筛选重复数据，提供参考，它对于提高数据质量和数据完整性有着极其重要的作用。传统的实体解析方法主要针对数据本身，如数据格式的统一、输入错误、缩写和数据中信息丢失等问题，大数据时代的到来，实体解析又面临着新的挑战：1）随着数据量的急剧增加，计算量也上升到一个新的高度，计算效率成为亟待解决的重大问题；2）数据之间的关系更加复杂，数据类型多样，且常出现不完整和不确定的数据，不仅要考虑实体与属性之间的关系，也要挖掘实体与实体之间的关系；3）不同领域数据的类型和格式均不相同，需要设计不同的实体识别方法对应不同的应用领域。

实体解析的一个标准方法是通过某种匹配方案计算两个实体属于同一个实体的可能性，这种方法的时间复杂度为 $o(n^2)$[2]，如何有效地减少计算量并同时保持解析的精度变得尤为重要。分块技术[3,4,5]正是解决该问题的有效方法。分块技术将众多实体按照一定标准划分成块，将实体之间的匹配限定在分块内部，时间复杂度由 $o(n^2)$ 减少为 $o(\omega\times n)$，极大地降低了计算量，提高了执行效率。但是，如果仅对块内实体进行识别，就会忽略块与块之间实体属于同一个实体的可能性。文献[6~8]中针对这个问题提出了解决方法。然而，随着大数据时代的到来，简单的分块方法无法保证大数据环境下实体解析的时间效率。

Hadoop[1]是一个分布式系统基础架构，它的核心是分布式文件系统 HDFS 和 MapReduce，HDFS 为海量数据提供分布式存储，MapReduce 为海量数据提供分布式计算。在 MapReduce 中，数据通过<key, value>键值对的形式在程序中进行传输和处理，用户只需定义一个 map 函数和一个 reduce 函数来进行相应计算：

$$map:(key_{in},value_{in})\rightarrow list(key_{tmp},value_{tmp})$$

$$reduce:(key_{tmp},list(value_{tmp}))\rightarrow list(key_{out},value_{out})$$

考虑到实体解析的数据解析格式以及计算量大的特点，MapReduce 编程模型非常适合于运用到实体解析中。图 1 为 MapReduce 下基于分块的实体解析流程。首先根据 key 值对原数据进行分块，将相同 key 值的实体划分到一个块中，之后对每一个块内的实体进行两两配对，再采用一种匹配策略对两两配对的实体对进行识别，将大于某个阈值的实体对挑选出来。

基于 MapReduce 框架的分布式解决方案是提高实体解析效率的有效方法，然而其本身也有一定的局限性，太过繁复的算法并不适合在 MapReduce 框架下实现。因此，尽管实体解析的方法技术已经相当成熟，但基于 MapReduce 框架实现的并不多。

本文中，我们基于 MapReduce 框架提出一种三阶段并行实体解析方法，该方法在分块的基础上，采用基于块依赖的数据筛选策略，将原始分块中与

[1] Hadoop: http://hadoop.apache.org/.

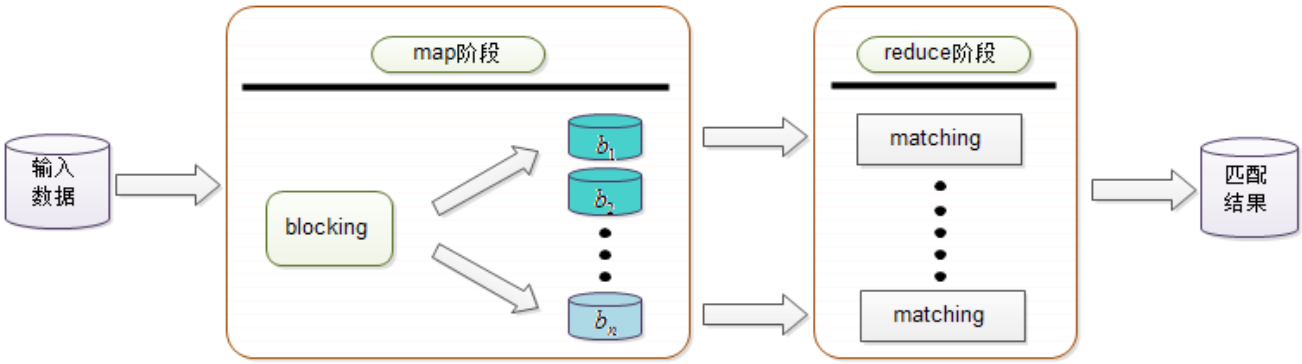

Fig. 1 General process of entity resolution in MapReduce
图 1 MapReduce 下基于分块的实体解析流程

所在块依赖度较低的实体挑选出来，与其他块中一定跨度距离的实体进行计算匹配。该方法通过分块和设定跨度距离来提高解析的时间效率，同时通过数据筛选来保持解析的有效性。

本文的主要工作及贡献如下：

1. 提出三层并行实体解析模型，在保证解析有效性的同时提高时间效率;

2. 提出基于块依赖的数据筛选策略，根据实体与块之间的依赖程度挑选进行块之间匹配的实体，在保证实体解析质量的同时减少计算量；

3. 设定跨度距离，有效调节块之间匹配的范围，在进一步控制计算量的基础上，提高解析效率；

4. 在 Hadoop 和真实数据集上的实验证明，基于块依赖的并行实体解析方法具有良好的解析质量和效率。

2.相关工作

2.1MapReduce

MapReduce 是一个可用于解决海量数据问题的编程模型，用户只需定义一个 map 函数和一个 reduce 函数来进行相应的计算。Map 函数和 reduce 函数均可在并行环境下执行，map 任务将输入数据转换成(key,value)对的形式，并将相同 key 值的数据聚集到一起进行输出。相同 key 值的数据将会分配到同一个 reduce 任务中进行处理。

Hadoop 是基于 MapReduce 编程模型的分布式架构，由于其开源且运行环境容易搭建，因此广泛应用于基于 MapReduce 编程模型的算法评估中。在 Hadoop 中，我们需要设置 map 的数量和 reduce 的数量，map 和 reduce 数量的变化将会影响程序的执行效率。通常，map 的数量是由输入数据的大小和 blocksize 的大小决定的:

$$number_{map} = inputsize / blocksize \quad (1)$$

Hadoop 有自己的任务调度机制，当一个 map 任务执行完成后，Hadoop 会将下一个 map 任务分配到当前空闲的节点上继续执行。程序执行所需的输入文件和输出文件可存储在 Hadoop 自己的文件系统 HDFS 上，HDFS 具有高效的数据读写吞吐率。

2.2 基于 MapReduce 框架的实体解析

实体解析最早是由 Fellegi 和 Sunter 于 1969 年在文献[9]中提出，时至今日，实体解析仍然是当今非常热门的研究课题[10]。大数据环境给实体解析带来了新的挑战。MapReduce 是一个可处理大数据集的编程框架，Kolb 等人基于 MapReduce 框架构建了一个针对大型数据集的实体解析系统 dedoop[11,12]，并采用高效的负载平衡策略来提高解析的时间效率[13]。众包[14]的方法将机器计算与人类智慧相结合，提高了解析效率和精度，其中 MapReduce 框架主要用于排除不可能匹配的记录对[15]。文献[16,17]将经典的 sorted-neighborblocking 分块方法在 MapReduce 上实现，用于提高解析的时间效率。该方法在初次分块后，针对每个相邻的分块，设定一个匹配参数 *window*，比较每一对相邻分块中相邻的 *window* 个实体。

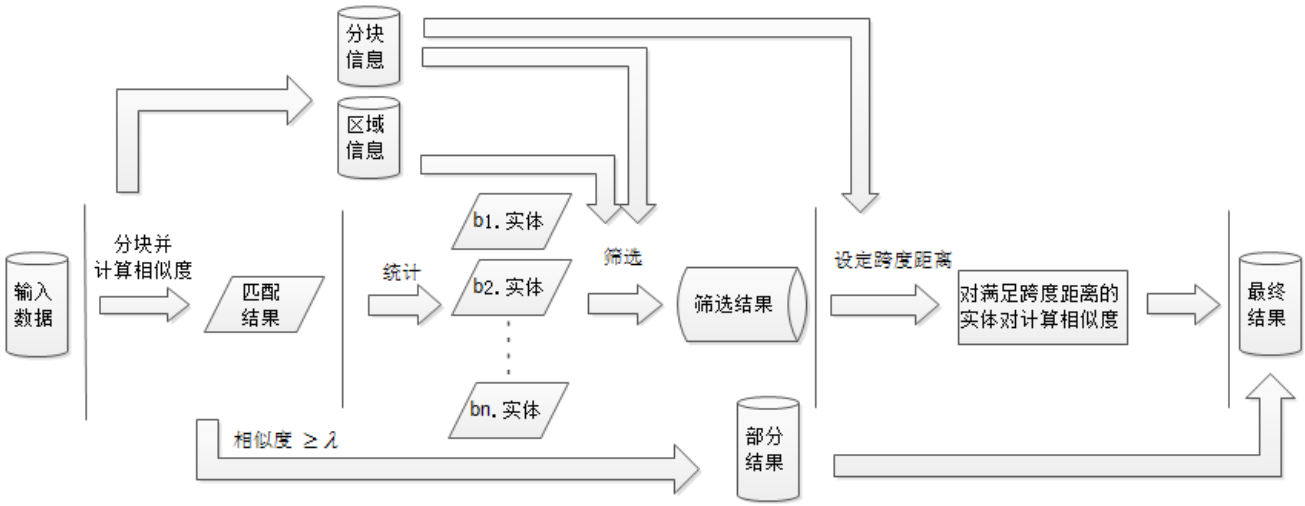

Fig. 2 Entity resolution process based on block-dependency

图 2 基于块依赖的实体解析流程

3.问题和定义

本文的讨论中，主要围绕三个问题展开：

问题 1：给定输入数据集 $E=\{e_1,e_2,...,e_n\}$，$n\le T$，T 为常数，对于 E 中实体所形成的实体对 $p=\{<e_i,e_k>|e_i,e_k\in E,i\neq k\}$，实体解析的任务是将实体对的相似度大于阈值λ（即 $sim(e_i,e_k)>\lambda$）的实体对挑选出来。

为了保证实体解析的精度和时间效率，需要引出问题 2 和问题 3。问题 2 通过数据筛选保证解析的精度，同时，它也能对计算量的控制起到一定作用；问题 3 通过设定跨度距离来调节计算量，保证时间效率。为了更好地理解问题 2 中的数据筛选，我们给出定义 1-定义 3。

定义 1 对于实体对 $<e_i,e_k>$，τ_u 表示相似度值的上限，τ_l 表示相似度值的下限，我们称 $sim(e_i,e_k)>\tau_u$ 的实体对所组成的部分为上限区域，记作 $upper\_limit(e_i,e_k)$；$sim(e_i,e_k)<\tau_l$ 所组成的部分称为下限区域，记作 $lower\_limit(e_i,e_k)$.

定义 2 对于块 b 的每一个实体 e，我们定义 e 对于 b 的块依赖度为 e 代表的实体在块 b 中出现的次数，记作 $block\_level(b,e)$。其中，e 在上限区域出现的次数为 e 对于 b 的上限块依赖度，记作 $bupper\_level(b,e)$；e 在下限区域出现的次数为 e 对于 b 的下限块依赖度，记作 $blower\_level(b,e)$。

定义 3 对于每一个分块中的上限区域和下限区域，δ_u 定义为上限区域中块依赖度的阈值，δ_l 定义为下限区域中块依赖度的阈值，其中

$$\delta_u=\frac{number_u-1}{2}\times\frac{number_u}{number_b} \quad (2)$$

$$\delta_l=\frac{number_l-1}{2}\times\frac{number_l}{number_b} \quad (3)$$

$number_u$ 表示上限区域中实体的总数，$number_l$ 表示下限区域中实体的总数，$number_b$ 表示该分块中实体的总数。

定义 1-3 主要针对问题 2—数据筛选，方便我们了解被筛选实体的所在位置和属性特点，数据筛选的任务在问题 2 中描述。

问题 2：对块 b 中上限区域和下限区域出现的实体对，数据筛选的任务是在 $upper\_limit(e_i,e_k)$ 中选出 $bupper\_level(b,e)\le\partial_u$ 的实体，同时在 $lower\_limit(e_i,e_k)$ 中选出 $blower\_level(b,e)\ge\partial_l$ 的实体。

第一阶段的分块并不能完全将相似的实体聚集到一起，分块中的实体可能并不属于该块。因此我们定义了区域和块依赖度来对实体进行筛选，尽可能将划分失误的实体筛选出来。块依赖度表示某个实体在块中出现的次数，它与块中实体的总数和

区域中实体的总数相关。以公式(2)为例，实体在区域中出现的次数最多为 $number_u$-1次，假设该实体未达到 $(number_u\text{-}1)/2$次，那么它极有可能被划分失误。因此我们设定的依赖度的阈值正好基于 ($number_u$-1)/2 这个平衡点。同时在公式(2)中加入了 $number_u / number_b$,该部分表示区域实体总数在整个块中所占的比例，这样定义的阈值既考虑了实体的分布，也考虑了局部区域内实体的出现情况。对于已筛选出的实体，我们将作第二次匹配计算，此时，我们需要设定跨度距离来控制计算量。

定义 4 对于块 b 中的实体，跨度距离定义为块 b 中两个实体之间间隔的实体数量，记作 sdistance。

第二次匹配时，对挑选出的实体，在将它与其它块中的实体配对时，我们仅对满足跨度距离条件的实体匹配，这样可以减少计算量，提高时间效率。

问题 3：对于数据筛选中筛选出的实体 e，跨度距离设定的任务是使该实体与其他分块中的实体按跨度距离 sdistance 进行匹配，即对分块 b 中的 $\{e_1,e_2,...,e_n\}$ ，仅对 $\{< e,e_{k+sdistance} > \mid e\notin b\wedge 0\le k\le n\wedge e_{k+sdistance}\in b\wedge e_k\in b\}$ 的实体对进行匹配计算。

4.基于块依赖的实体解析方法

4.1 总体框架和流程

基于块依赖的实体解析方法主要从三个方面减少计算量：1）数据分块，将实体的计算限定在每一个分块内部；2）数据筛选，对于已计算相似度的各分块内的实体对，通过划分上限区域和下限区域筛选实体，上限区域和下限区域块依赖度的阈值可决定筛选出的实体的数量；3）对于筛选出的实体，设定跨度距离，仅对其他每一个分块中（不考虑筛选出的实体最初的所在块）符合距离要求的实体进行匹配，因此可以通过调节跨度距离的大小来控制计算量。此外，数据筛选是基于块依赖筛选实体，它同时也影响匹配结果的正确性。本文中，匹配结果的准确性主要由数据筛选和匹配算法决定，我们采用 Jaccard 算法作为匹配算法来计算两个实体之间的相似度。

图 2 为块依赖实体解析框架的三阶段运行流程：第一阶段对输入数据按照设定的 key 值进行分块并计算每一个块内实体对的相似度，提取分块信息和区域信息为后续阶段使用，同时将其中相似度值大于阈值λ的实体对判定为相同实体并作为最终结果的一部分输出；第二阶段对第一阶段的全部实体对按照所划分的块进行统计，并根据分块信息和区域信息进行计算筛选，挑选出每一个块中满足筛选条件的实体；第三阶段通过设定距离跨度，将筛选出的实体与其他块中满足跨度距离的实体进行再匹配。

4.2 基于块依赖的数据筛选

第一阶段的分块方式简单且能有效减少实体对的数量。但是，简单的分块方法必然会导致数据容错性较高，划分到一个分块中的实体不一定与该块中的实体有较高的匹配度。例如我们按照姓名的首字母进行分块，对于“Shapire, r”，若表达方式不同，一个将数据存为“shapire, r”，另一个存为“R, shapire”，那么他们将会被划分到不同的分块中，这样的结果是我们不愿意见到的，因此我们希望对块与块之间的数据再作一次比较，如何选择待比较的数据便是需要考虑的问题。

从上述例子中，我们可以发现，两条记录中必然会有一条记录与它所在块中的实体的匹配度较低，即在下限区域中，这一条记录的 blower_level 较高。因此我们考虑将满足这个条件的实体筛选出来。

此外，在上限区域中也存在一种可筛选的情况。即存在两个实体，他们的匹配度较高，但是他们与同样排名靠前的实体的匹配度却较低，即他们在上限区域的实体序列中出现的次数较少，这说明这两个极有可能属于同一个实体的记录恰好被分配到了一个分块中，他们彼此拥有很高的匹配度但是他们不属于这一个分块。

数据筛选主要是为了保证解析的精度，但它对于计算量的减少也有一定的贡献。相似度值的上限

和下限可以决定上限区域和下限区域的大小，即可决定上限区域和下限区域中包含的实体的数量，也就会导致上限区域和下限区域中块依赖度的阈值的变化，最终影响筛选出的实体的数量，从而可以改变计算量。为了实现 MapReduce 下基于块依赖的数据筛选方法，我们约定数据筛选阶段数据的输入格式和输出格式如下：

<块 id/区域 id/实体 id1，块 id/区域 id/实体 id2>

<块 id，实体 id>

数据筛选的算法如图 3 所示。Map 中通过交换输入实体对形成中间数据，不同的 value 值用于区别和统计实体（2~3 行）。进行 reduce 之前，先从缓存中获得块信息和区域信息（4~7 行），之后进入 reduce 函数，首先计算实体出现次数（10~11 行），获得区域实体总数和块中实体总数（13~15 行），再计算块依赖度（16 行），最后通过判断区域号识别上限区域和下限区域，再进行判断后输出（17~22 行）。

4.3 跨度距离设定

分块和距离设定都能减少匹配的实体对数量，提高时间效率。第一阶段，我们将对数据按照设定的 key 值进行分块，将相同 key 值的记录划分到一个分块中。第二阶段通过上限区域和下限区域的上限块依赖度和下限块依赖度挑选在第一阶段中可能划分失误的实体于第三阶段中进行匹配，尽可能弥补第一阶段遗漏的可以匹配的实体对。我们称第三阶段中的匹配为二次匹配。

二次匹配主要通过设定跨度距离来调节匹配数量，以便控制计算量，在保持有效性的基础上提高时间效率。跨度距离是指同一个分块中两个实体之间间隔的实体数量。在进行二次匹配时，对于筛选出的实体，我们将它与其它块中的实体通过跨度距离来匹配。第三阶段的解析算法如图 4 所示。

首先通过 map 的 setup 函数获得分块的相关信息（1~3 行），之后在 map 中设定二次匹配的匹配范围 C $(0 \le C \le 1)$，并计算跨度距离在实验中的数值（5~6 行）。C 为人为设定的可调节的匹配范围，sdistance=block_size/ block_size *C = 1/C.在获

算法 1 数据筛选算法

```
1.  map(key_in =b_id/z_id/e_id1, value_in =b_id/z_id/e_id2){
2.    output(key_tmp =b_id/z_id/e_id1,value_tmp=e_id2);
3.    output(key_tmp=b_id/z_id/e_id2,value_tmp= e_id1);}
4.  reduce_setup(jobconf){
5.    local_files[]←getfromCache(files);
6.    list(list(block))←get_file(local_file[0]);
7.    list(list(z_number))←get_file(local_file[1]);}
8.  reduce(key_tmp = b_id/z_id/e_id, list(value_tmp)=
    list(o_e_id)){
9.    z_count←0;    z_level←0;
10.   for each o_e_id in list(o_e_id)){
11.     z_count ++;}
12.   infor[]←key_tmp.split("/");
13.   b_id←infor[0];
14.   b_size←list(list(block)).get(b_id).size;
15.   z_size←list(list(z_num)).get(b_id).get(z_id);
16.   z_level←(z_size-1)*z_size/2*b_size);
17.   if z_id == 0 then{
18.     if z_count < z_level ‖ z_count == z_level
19.       then output(b_id, e_id);}
20.   if z_id == 1 then{
21.     if z_count > z_level ‖ z_count == z_level
22.       then output(b_id, e_id);}}
```

Fig. 3 Data screening algorithm

图 3 数据筛选算法

算法 2 跨度距离设定算法

```
1.  map_setup(jobconf){
2.    local_file←getfromCache(B_file);
3.    list(list(block))←get_N(local_file);}
4.  map(key_in = b_id, value_in =e_id){
5.    range ← C;
6.    d←1/range;
7.    for each list(block) in list(list(block)){
8.      if it is not the list(block) of b_id then{
9.      b_size←list(block).size;
10.     count←0;
11.     while count < b_size{
12.     p_e←list(list(block)).get(b_id).get(count);
13.     output(key_tmp = entity, value_tmp =p_e);
14.     count←count+d;}}}}
15. reduce(key_tmp =entity, list(value_tmp) = list(p_e)){
16.   for each picked_entity in list(p_e){
17.     sim←compute_sim(entity,picked_entity);
18.     If sim > λ then output(entity,picked_entity);}
```

Fig. 4 Span distance setting algorithm

图 4 跨度距离设定算法

得跨度距离值后，根据跨度距离匹配实体（11~14 行），并在 reduce 中进行相似度计算，将大于阈值λ的实体对挑选出来（15~18 行）。

5.实验和评估

5.1 实验环境和数据

实验部署的分布式环境由 4 个节点构成，其中 2 个节点（1 台为主节点）为内存 16G，CPU 2.93GHZ，硬盘 1T 的 Dell 服务器，另外 2 个节点为内存 4G，CPU2.73GHZ，硬盘 1T 的 Dell 服务器。实验所用的算法是在 Hadoop2.20.0 和 jdk1.7.0_06 环境下实现的.另外我们对 Hadoop 的相关设置作了以下调整：我们将 block size 的大小设置为 128MB，master 节点既进行 task 的调度工作，同时也作为 slave 节点执行相关的运算任务。每一个节点在并行情况下最多执行两个 map 任务和 reduce 任务。

我们使用 DBLP-scholar 和 citeseer 数据集作为实验数据。DBLP-scholar 数据集共两个文件，文件 1 包含 65000 条学术论文，文件 2 包含约 2500 条学术论文，共 $65000*2500 \approx 1.6*10^9$ 个数据对。DBLP-scholar 拥有标准的匹配准则，因此用它来验证实验结果的有效性。citeseer 数据集拥有 120 多万条数据记录，共约 $120*10^4*(120*10^4-1)/2 \approx 7.2*10^{11}$ 个记录对。本文随机选择其中 10 万条数据共 $1*10^5*(1*10^5-1)/2 \approx 5.0*10^9$ 个数据对进行时间效率的评估实验。

5.2 实验评估方法

本文主要针对实验的有效性和时间效率进行评估。准确率、召回率和 F-值是常用于评估实体解析的三个指标，我们用其评估算法的有效性。此外，基于块依赖的实体解析主要是在保持有效性的基础上减少计算量，因此我们将在实验中讨论有效性与计算量之间的关系。

设 $pairs\_right$ 表示实验中实际正确的实体对数目，$picked\_pairs\_right$ 表示实验中正确识别出的实体对数目，$pairs\_total$ 表示实验中识别出的实体对总数。准确率（precision）、召回率(recall)和 F-值(F-measure)的计算公式如下：

$$\text{precision} = \frac{\text{picked\_pairs\_right}}{\text{pairs\_total}} \tag{4}$$

$$\text{recall} = \frac{\text{picked\_pairs\_right}}{\text{pairs\_right}} \tag{5}$$

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

由于实验在分布式环境下进行，因此我们通过加速比来评判时间提高的幅度。加速比是指同一个任务在单处理器系统和并行处理器系统中运行消耗的时间比率，用来衡量并行系统或程序并行化的性能和效果。在 MapReduce 框架下，map 的计算工作分布在多个节点中并行执行，加速比常用来衡量单节点和多节点下算法的执行性能。

$$\text{speed\_up} = \frac{T_1}{T_p} \tag{7}$$

其中 T_1 表示单处理器下的运行时间，T_p 表示 p 个处理器下的运行时间。

5.3 实验结果与分析

5.3.1 有效性

实验运行在 4 个节点上，由于每一个节点是双核的，因此 4 个节点共可运行 8 个 map 任务，设置 reduce 的任务数量为 1。以下是二次匹配时的范围设定和对应的跨度距离，我们共设定了 13 个匹配范围，其值和对应的跨度距离如下：

C={0,0.2%,0.33%,0.8%,1%,1.2%,1.4%,2%,2,5%,4%,6.7%,8.3%,10%}

sdistance={|b|,500,300,125,100,80,70,50,40,25 25,15,12,10}

当匹配范围为 0 时，跨度距离表现为块的大小，用|b|表示。

此外，设定 $\lambda = 0.75, \tau_u = 0.85,\ \tau_l = 0.25$[2]。

[2] λ的设定参考论文[17,18]，τ_u 和 τ_l 的设定主要为了控制实体对数量，可调整。

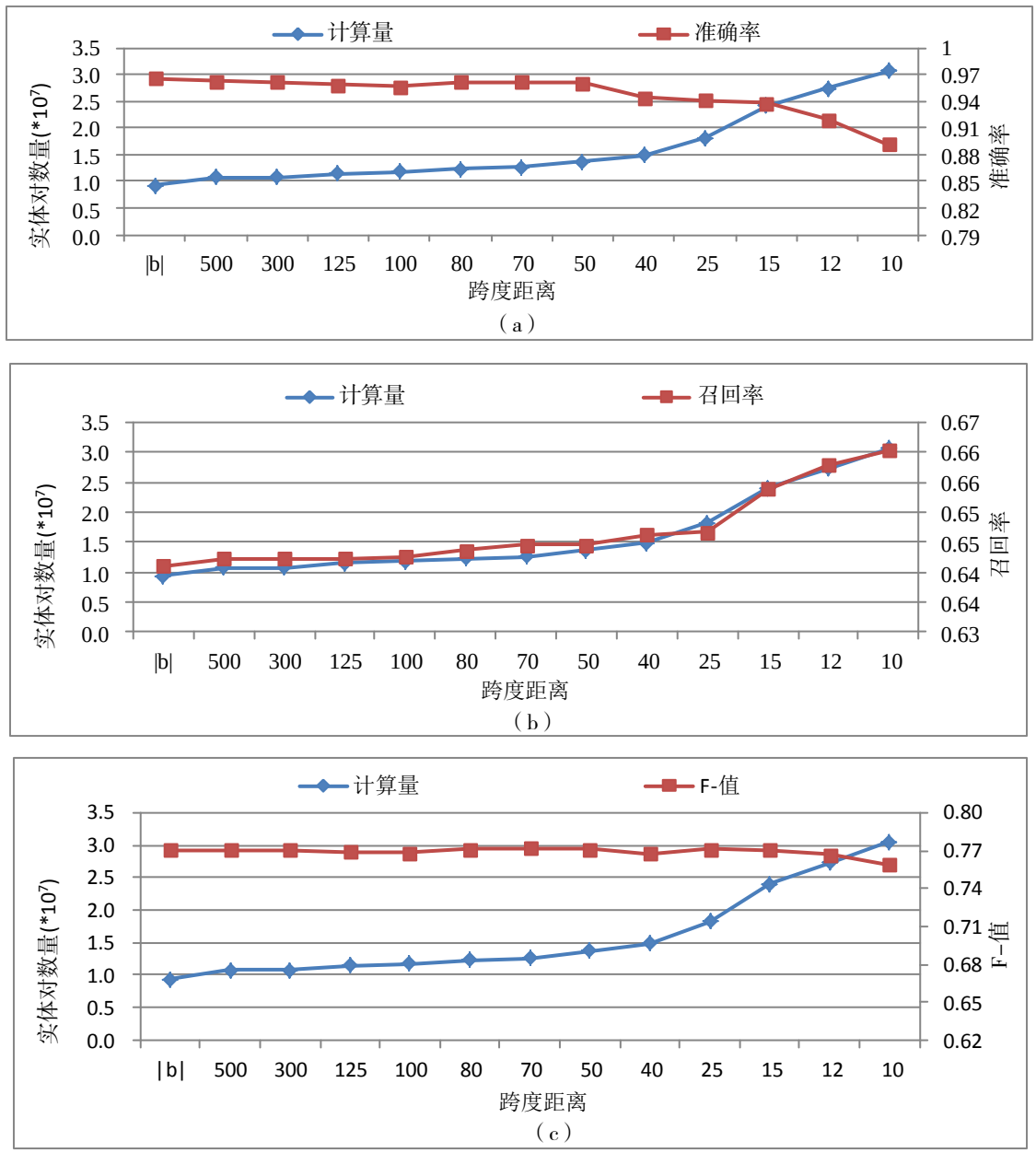

Fig. 5 Relationship between span distance and the amount of calculation、precision, recall and F-measure

图 5 跨度距离与计算量、准确率、召回率和 F-值的关系

图 5 为跨度距离与准确率、召回率、F-值以及计算量之间的关系。从图中可看出:(1)对于图 5(a),随着跨度距离的减小,准确率的总体发展轨迹是向下的,但在跨度距离为 80~50 时有一个向上的增长趋势,而在跨度距离为 25 以后,准确率曲线向下的倾斜程度变大,此时计算量急剧增加,说明跨度距离越小,计算量越大,准确率减少的速度越快;(2)图 5(b)显示,随着距离的减小,召回率一直保持增长的趋势,且它在跨度距离为 15 时的增长率最大,此后,计算量大幅增加,召回率的增长速度相对跨度距离为 15 时变慢;(3)F-值由准确率和召回率同时决定,从图 5(c)可看出,F-值基本保持在 0.77 左右,在跨度距离为 70 时,F-值最高,随着距离减小,F-值呈现缓慢下降的趋势;(4)随着跨度距离的减小,计算量增大,当距离减小到一定程度时,跨度距离值很小幅度的减少,都会导

致计算量的急剧增加。

对准确率、召回率和 F-值的变化趋势作进一步的分析发现：

（1）准确率由正确识别的实体对数目和识别出的实体对数目决定，当距离减小时，计算量增大，导致干扰的实体对数目增多，使得识别出的总实体对数目增加，尽管正确识别的实体对数目也会增加，但增加的数量远不如干扰项的增加量多，因此准确率总体会呈现下降趋势；

（2）召回率由正确识别的实体对数目和实际正确的实体对数目决定，当距离减小，计算量增加，使得正确识别的实体对数目增加，因此召回率呈增长趋势；此外，二次匹配的实体对由数据筛选和跨度距离决定，数据筛选是有依据地挑选实体，但跨度距离的值是人为决定的，因此二次匹配的实体对有一定的随机性，且实际正确的实体对数目固定，因此，当计算量增大时，正确识别的实体对数目的增加速度会变慢，召回率的增长也会减慢；

（3）随着距离的减小，准确率下降的速度加快，且其减少的程度高于召回率增加的程度，因此，F-值会最终呈现下降趋势；

（4）跨度距离的设定与块内实体数及数据筛选所挑选的实体总数有关系，对于实验采用的数据集，当跨度距离为 80~50 之间时，准确率和 F-值都达到最优的结果，二次匹配的范围在 1.2%~2%之间时达到最优。

表 1 为各跨度距离下的实际计算量、实际计算量占总量的百分比以及减少率。为方便显示，表 1 中未列出跨度距离为 70 和 40 的情况。从表 1 可看出，跨度距离的设定能很好地控制实际计算量，且在有效性最优的情况下，计算量的减少率到达 90%以上。

综上所述，基于块依赖的实体解析方法通过数据筛选和设定跨度距离，在保证解析精度的情况下大幅减少计算量，以期获得较好的时间效率。

5.3.2 时间效率

MapReduce 框架能够处理大数据集的运算工作，因此我们采用它来对基于块依赖的实体解析方法进行时间效率的评估。图 6 是总运行时间随 map 数增长变化的图例，此外我们也对该实验进行了加速比的评估。实验采用 4 个节点，map 的数量最多可达 8 个，reduce 数量设置为 1，距离 d 设置为 100。

程序的总运行时间由三个阶段各自的运行时间构成，其中数据筛选部分所占比例最少，分块阶段占据了大部分的运算时间。图 5 显示，随 map 数量的增加，程序的总运行时间随之降低，这说明我们的算法能很好地适应分布式环境。

然而由于 reduce 数量设置为 1，每一阶段 reduce 的工作仅由一个 reduce 任务完成而无法并行处理，且第一阶段和第三阶段的 reduce 任务包含了对实体进行两两配对和相似度计算两部分，这使得 reduce 的效率降低，也使得算法总运行时间的加速比效果不佳。于是，我们考虑将 reduce 的计算任务单独放在一个 job 中完成，使得计算部分能在多个 map 中并行处理，发现该部分程序运行的加速比达到了一个非常满意的效果，如图 7 所示。

Table 1 The amount of calculation under various span distances

表 1 各跨度距离下的计算量

跨度距离	|b|	500	300	125	100	80	50	25	15	12	10
计算量 ($*10^7$)	0.93	1.07	1.08	1.15	1.18	1.23	1.37	1.82	2.41	2.73	3.06
实际计算百分比	5.5%	6.4%	6.4%	6.8%	7.0%	7.3%	8.2%	10.8%	14.3%	16.3%	18.2%
相对总量的减少率	94.5%	93.6%	93.6%	93.2%	93.0%	92.7%	91.8%	89.2%	85.7%	83.7%	81.8%

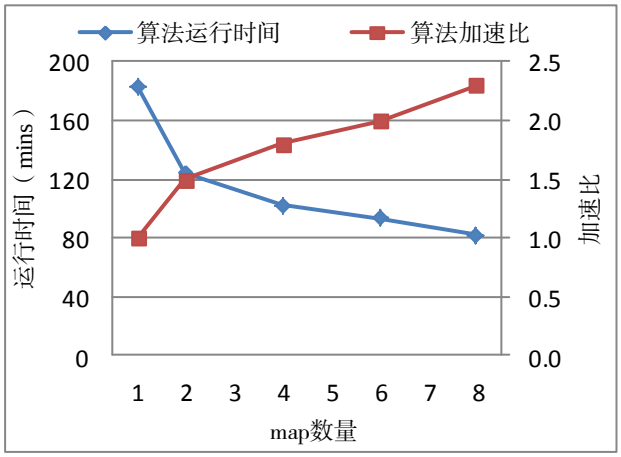

Fig. 6The variation tendency of reduce time and speed up with map tasks

图 6 reduce 运行时间和加速比随 map 数变化的趋势

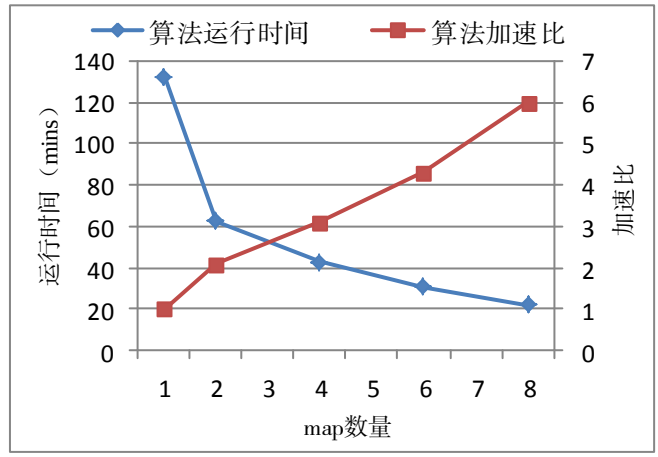

Fig. 7 The variation tendency of total time and speed up with map tasks

图 7 总时间和加速比随 map 数量变化的趋势

从理论上讲，加速比应与 map 数量成正比，但由于要考虑节点间的数据传输以及 map 和 reduce 任务初始化等一些因素，导致加速比低于理论值。从图 7 中可看出，该部分运行时间的加速比基本呈线性增长。

距离设定越小，reduce 的计算量则越大。图 8 为 reduce 计算任务运行时间与距离的关系的图例。实验运行在 4 个节点的环境下，map 数量设置为 8，reduce 为 1。随距离值的增长，运行时间呈缓慢的下降趋势。当距离值为 50、200、400 时，运行时间呈线性的下降状态，这种情况产生的原因是由于后一个距离值的设定恰好导致程序的总 map 任务数比前一个距离值下的 map 任务数恰好减少了 8 的倍数。这说明，距离的设定能有效减少计算量，从而减少程序的运行时间。

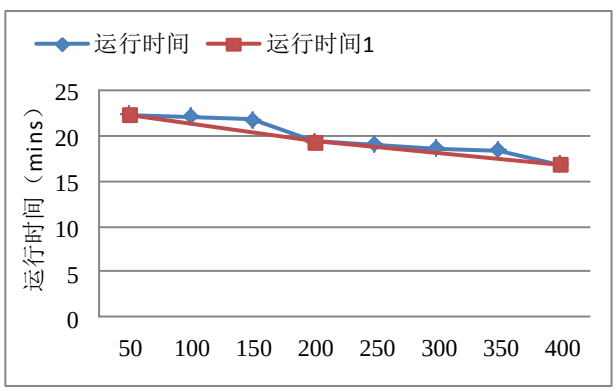

Fig. 8 The relationship between reduce task time and span distance

图 8 reduce 计算任务运行时间与距离的关系

6.总结与未来工作

大数据环境下，实体解析的方法对时间效率的要求越来越高，但同时解析结果的正确性也是不容忽视的因素。本文基于 MapReduce 框架实现了一种基于块依赖的实体解析方法，通过数据筛选挑选出与所在块的依赖度较低的实体与其他块的实体再进行匹配，在保证解析正确性的基础上，减少计算量，同时采用跨度距离设定进一步减少计算量，提高时间效率。

今后我们将考虑采用加载平衡策略，使用多个 reduce 任务来并行处理，以便更大程度地提高程序的时间效率，同时改进相似度计算方法，提高解析的有效性。

References

[1] Getoor L, Machanavajjhala A. Entity resolution for big data[C]//Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chicago, USA, August 11-14,2013 ACM New York, NY, USA 2013:1527-1527

[2] Hanna, K., Thor, A., Erhard R. Evaluation of entity resolution approaches on real-world match problems[J]. Proceeding of the VLDB Endowment, 2010, 3(1-2):484-493..

[3] Baxter R, Christen P, Churches T. A comparison of fast blocking methods for record linkage[C]//Proceedings of the 9th annual ACM SIGKDD Conference. 2003,3:25-27.

[4] Papadias G, Loannou E, Niederee C, et al. Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data[C]//Proceedings of the fifth ACM international conference on Web search and data mining. ACM New York, NY, USA, 2012:53-62.

[5] Michelson M, Knoblock C A. Learning blocking schemes for record linkage[C]//Proceedings of the National Conference on Artificial Intelligence. Menlo Park, CA, Cambridge, MA, London,. AAAI Press, MIT Press, 2006, 21(1):440

[6] Whang S E, Menestrina D, Koutrika G, et al. Entity resolution with iterative blocking[C]// Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. ACM New York, NY, USA, 2009:219-232.

[7] Bilenko M, Kamath B, Mooney R. Adaptive blocking:learning to scale up record linkage[C]// Sixth International Conference on Data Mining(ICDM '06). IEEE, 2006:87-96.

[8] Herzog T N., Scheuren F J, Winkler W E. Data quality and record linkage techniques[M]. Springer New York, July 2007.

[9] Fellegi I P, Sunter A B. Theory for record linkage[J]. Journal of the American Statistical Association, 1969, 64(328):1183-1210.

[10] Yan Canrong, Wan Yongquan. Parallel Entity Resolution and Record Aggregation Model[J]. Journal of Chinese Computer Systems. 2013, 34(8).

[11] Kolb L, Rahm E. Parallel entity resolution with Dedoop[J], Datanbank-Spectum 2013,13(1): 23-32.

[12] Kolb L, Thor A, Rahm E. Dedoop: efficient deduplication with Hadoop[J]. Proceedings of the VLDB Endowment, 2012,5(12):1878-1881.

[13] Kolb L, Thor A, Rahm E. Load balancing for mapreduce-based entity resolution[C]// Proceedings of 28th International Conference on Data Engineering(ICDE), 2012, IEEE, 2012 : 618-629.

[14] Wang J, Kraska T, Franklin M J, et al. Crowder: Crowdsourcing entity eesolution[J]. Proceedings of the VLDB Endowment, 2012, 5(11):1483-1494.

[15] Yan Cairong, Zhang Yangshun, Xu Guangwei. Crowdsourcing entity resolution with privacy protection[J]. Journal of Frontiers of Computer Science & Technology. 2014, 8(7):802-811.

[16] Kolb L, Thor A, Rahm E. Parallel sorted neighborhood blocking with mapreduce[J]. arXiv preprint arXiv:1010.305, 2010: 45-64.

[17] Kolb L, Thor A, Rahm E. Multi-pass sorted neighborhood blocking with mapreduce[J]. Computer Science-Research and development, 2012, 27(1):45-63..

[18] Whang S E, Garcia-Molina H. Joint Entity resolution[C]//Proceedings of the 28th International Conference on Data Engineering. IEEE, 2012:294-305.

参考文献

[10] 燕彩蓉，万永权.并行实体解析与记录聚合模型[J]. 小型微型计算机系统. 2013, 34(8)：1843-1847.

[15] 燕彩蓉，张洋舜，徐光伟.支持隐私保护的众包实体解析[J]. 计算机科学与探索,2014, 8(7):802-811.

Wang Ning was born in 1967. She received the Ph.D. degree from Southeast University in 1998. Now she is an associate professor at Beijing Jiaotong University, and member of China Computer Federation. Her research interests include Web data management, data mining and information retrieval, etc.

王宁（1967—），女，江苏常州人，1998 年于东南大学获得博士学位，现为北京交通大学副教授，CCF 会员，主要研究领域为 Web 数据管理，数据挖掘，信息检索等。

Huang Min was born in 1989. She is postgraduate at Beijing Jiaotong University. Her research interests include web data management and data integration, etc.

黄敏（1989-）女，四川成都人，北京交通大学硕士研究生，主要研究领域为 web 数据管理和数据集成等。