RDF数据分布式并行语义编码算法

郑翠春 汪璟玢

(福州大学数学与计算机科学学院 福州 350108)

摘 要 现有的RDF数据分布式并行压缩编码算法均未考虑结合本体文件,导致编码后的RDF数据没有表示任何语义信息,不利于分布式查询或推理。针对这些问题,提出SCOM(Semantic Coding with Ontology on MapReduce)算法在分布式MapReduce下完成RDF数据的语义并行编码。该算法首先结合RDF数据本体,构建类关系和属性关系模型;在三元组项分类与过滤之后,对三元组项进行编码并生成字典表,最终完成RDF数据带有语义信息且具有规律性的编码。此外,SCOM算法能够很容易地将编码后的RDF数据文件恢复为原始文件。实验表明,SCOM算法能够高效地实现大规模数据的分布式并行编码。

关键词 RDF,本体,语义编码,MapReduce

中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.9.039

Distributed Parallel Semantic Coding Algorithm for RDF Data

ZHENG Cui-chun WANG Jing-bin

(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China)

Abstract The existing distributed parallel compression coding algorithms for RDF data do not consider combining with the ontology file, resulting in encoded RDF data without any semantic information, which is not conducive to the distributed query or reasoning. To solve these problems, a method named SCOM (Semantic Code with Ontology on MapReduce) was proposed to complete the semantic parallel coding for RDF data. Firstly, the algorithm combines the ontology of RDF data to build the class and attribute relationship model. The triple items are encoded and a dictionary table is generated after classifying and filtering triples. Finally, the coding for RDF data with semantic information and regularities is completed. In addition, SCOM algorithm can easily revert the encoded RDF data file to their original file. Experimental results show that SCOM algorithm can achieve the parallel coding of large-scale data efficiently.

Keywords RDF, Ontology, Semantic coding, MapReduce

1 引言

RDF(Resource Description Framework)是由WWW提出的对万维网(World Wide Web)上的信息进行描述的一个框架[1]。随着语义Web技术的迅速扩展,RDF数据格式应用到越来越多的领域,如一般知识(DBpedia[2])、生物信息学(UniProt[3])和地理信息系统(Linkedgeodata[4]),截至2014年9月,数据的总量已达650亿条三元组。RDF数据的大规模性,使得其管理存在局限性。为了加速RDF数据的查询或推理并缩小数据的存储空间,通常的做法就是对三元组进行压缩编码。压缩编码已经被证明是一种有效的编码,通过用一个数值(ID)替换原有的三元组项(主语或谓语或宾语),最终将所有三元组数据转换为数值式的数据。集中式环境由于内存的限制,不适用于对大规模数据的编码。研究RDF数据分布式并行压缩编码算法是目前较新的一个领域。Goodman等人[5]在Gray XMT机器上提出适应线性探测的方法,其通过并行的Hash在单个字典表上实现了并行的编码。此算法的编码时间与所采用的计算机核数呈线性关系,这种方法要求所有的数据保存在内存中,同时严重依赖于在共享内存架构的Gray XMT,不适用于普通的分布式存储系统。Long Cheng等人[6,7]采用X10语言对RDF数据进行压缩,首先对三元组进行过滤,再根据三元组项的Hash值将三元组数据等数量地分配到不同的节点进行本地编码,并生成多个字典表。Urbani等人[8]提出分布式MapReduce数据压缩算法,其主要分为数据压缩阶段和数据反转阶段,其中在数据压缩阶段对三元组进行压缩,并构建字典表;在反转阶段,将压缩后的三元组和字典表进行连接,从而生成原始三元组数据。此算法在数据反转阶段的效率不够高。

文献[5-8]为目前RDF数据分布式并行压缩的最新研究成果,也是现有的3种有效的RDF数据并行压缩算法,能够实现海量RDF数据并行压缩编码,但这些压缩算法均未考虑结合本体文件,因此编码后的三元组没有表示任何语义信息,

到稿日期:2015-05-08 返修日期:2015-08-15 本文受国家青年基金项目(61300104),福建省科技拥军基金项目(JG2014001),福建省自然科学基金项目(2012J01168),福州大学科技发展基金资助项目(2013-XQ-32)资助。

郑翠春(1989—),女,硕士生,主要研究领域为海量数据管理、智能技术,E-mail:software_cui@sina.com;汪璟玢(1973—),女,硕士,副教授,CCF会员,主要研究领域为海量数据管理、网络数据库和智能技术,E-mail:wjbcc@263.net(通信作者)。

不利于后期进行分布式查询或语义推理。为了解决上述压缩编码算法存在的问题，本文提出了 SCOM 算法（Semantic Coding with Ontology on MapReduce），结合本体对 RDF 数据进行编码，使得 RDF 三元组的编码带有语义信息且具有规律性，利于分布式查询与语义推理的完成。该算法首先读取本体文件构建类关系和属性关系模型，然后读取 RDF 数据文件，对三元组项进行分类与过滤，接着对三元组项进行编码，同时生成字典表；结合字典表，算法能够对编码后的 RDF 数据文件进行快速反转，生成原始的 RDF 数据文件。最后通过对比实验验证了 SCOM 算法在大数据量的情况下压缩编码与反转的效率较高。

本文第 2 节介绍 RDF 数据模型、MapReduce 模型与相关定义；第 3 节详细论述 SCOM 算法；第 4 节给出实验结果并将其与最先进的分布式压缩算法进行对比分析；最后总结全文。

2 基本概念

2.1 RDF 数据模型

RDF 的基本思想是通过统一资源标识符（Uniform Resource Identifier，URI）来标识 Web 上的资源，用简单的属性（property）以及属性值（value）来描述资源。RDF 模型可以采用不同的语法形式来描述。大多情况下都是用 XML 来刻画 RDF，其特点是简单、易于掌握和使用[9]。

三元组是 RDF 模型的另外一种常见描述方式。每一个三元组包括一个主语（subject）、一个谓语（predicate）和一个宾语（object）。陈述的主语（subject）通常是指向相应资源（resource）的 URI；谓语（predicate）是表示某一属性（property）的 URI；而宾语（object）既可以是指向某一个资源的 URI，也可以单纯地以一个文字（literal）作为属性值（property value）。除此之外，主语和宾语也可以是没有 URI 表示的匿名资源，亦称空白节点（blank node）[10]。本文采用以三元组形式表示的 RDF 数据。

除了以上两种方式之外，RDF 还可以用有向图的形式描述。一个 RDF 图的结点就是它包含的所有三元组的主语和宾语，而边的方向总是指向宾语[11]。在 RDF 图建立完成之后，便可通过图中内容来了解 RDF 所描述的资源的信息。基本的 RDF 图的表示形式如图 1 所示。

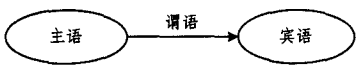

图 1 包含一个三元组的 RDF 图

2.2 MapReduce 模型

MapReduce 是一种并行的分布式处理框架，每个作业包括两个处理阶段：Map 阶段和 Reduce 阶段。Map 阶段会根据某个元素的键值对（key/value）对输入数据进行划分；Reduce 阶段将相同的 key 进行合并后产生输出结果。本质上，Map 阶段和 Reduce 阶段可以并行化，但在一个作业中可以先执行 Map 阶段后执行 Reduce 阶段[11]。

执行一个 MapReduce 程序需要 5 个步骤：输入文件、将文件分配给多个 worker 并行地执行、写中间文件、多个 Reduce workers 同时运行、输出最终结果。如图 2 所示。

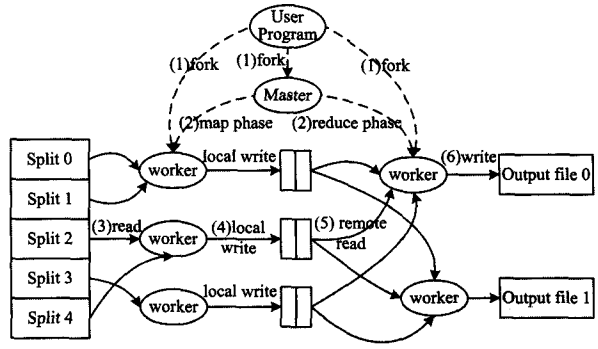

图 2 MapReduce 模型

2.3 相关定义

定义 1 三元组项（TripleItem）指三元组的主语或者谓语或者宾语。

$\forall(S_i,P_j,O_k)(1\leqslant i,j,k\leqslant n)$

其中，n 表示三元组的总数。若 $\forall v\in\{S_i,P_j,O_k\}$，则

$v\in TripleItem$

定义 2 类属性类型标记（Flag），用于标识类和属性。假设当前数据为 v，则

$$Flag=\begin{cases}0, & v\in\text{类}\\ 1, & v\in\text{属性}\end{cases}$$

定义 3 树节点编码位数（TreenodeDigit），简称 TD。若总节点数为 M，则

$$TD(M)=\begin{cases}1, & 0<M<10\\ TD(floor(M/10))+1, & M\geqslant 10\end{cases}$$

定义 4 类编码（TreeClasscode，TC）。结合定义 2 和定义 3，TC 由 Flag、直系父类个数标记（ImmediateParentnumFlag，IPF）、父类节点顺序编码和节点顺序编码构成。其中，总节点数为 M，父类节点顺序编码的位数与节点顺序编码的位数都为 $TD(M)$。$TC(h,i)$ 表示第 h 层的第 i 个节点 A 的类节点编码；$f(h,i)$ 表示第 h 层的第 i 个节点 A 的节点顺序编码，$REPT(0,n)$ 表示产生 n 个 0；设 $anc(h)$ 表示第 h 层的类节点顺序编码，$f(h-1,m)$ 表示节点 A 的父类节点 B 的节点顺序编码，则

$$f(h,i)=\begin{cases}\max\{anc(h-1)\}+i, & h\geqslant 1\\ 0, & h=0,i=1\end{cases}$$

$$TC(h,i)=Flag\&IPF\&REPT(0,TD(M)-TD(f(h-1,m)))\&f(h-1,m)\&REPT(0,TD(M)-TD(f(h,i)))\&f(h,i)$$

当 $IPF>1$ 时，父类节点顺序编码为所有直系父类的节点顺序编码的组合。

定义 5 属性编码（TreePropertycode，TP）。结合定义 2—定义 4，TP 由 Flag、类编码、父属性节点顺序编码和节点顺序编码构成。其中，总节点数为 M，父属性节点顺序编码的位数与节点顺序编码的位数都为 $TD(M)$。$TP(h,i)$ 表示第 h 层的第 i 个节点 C 的属性节点编码，C 所属的类设为 R，其类节点编码表示为 $TC(p,r)$；$f(h,i)$ 表示第 h 层的第 i 个节点 C 的节点顺序编码，$REPT(0,n)$ 表示产生 n 个 0；设 $anc(h)$ 表示第 h 层的属性节点顺序编码，$f(h-1,m)$ 表示节点 C 的父属性节点 D 的节点顺序编码，则

$$f(h,i)=\begin{cases}\max\{anc(h-1)\}+i, & h\geqslant 1\\ 0, & h=0,i=1\end{cases}$$

$TP(h,i)=Flag\&TC(p,r)\&REPT(0,TD(M)-TD(f(h-1,m)))\&f(h-1,m)\&REPT(0,TD(M)-TD(f(h,i)))\&f(h,i)$

3 SCOM 算法

SCOM 算法根据 MapReduce 的特点，结合本体构建类关系和属性关系模型，根据模型对 RDF 数据进行分类编码，从而能够实现 RDF 数据的分布式并行压缩编码。SCOM 算法分为压缩编码阶段和反转阶段，主要包括以下几个步骤：

Step1　读入本体文件，构建类关系和属性关系模型，生成类(属性)及其编码的映射文件。

Step2　为了确保 RDF 三元组编码的一致性，使得同一个 TripleItem 不会分配到不同的编码，需要对 TripleItem 进行过滤。读入 RDF 数据文件，将三元组分割成 TripleItem 且将 TripleItem 按类划分，并删除重复的 TripleItem，同时生成前缀编码。

Step3　TripleItem 编码，生成字典表。

Step4　三元组的编码，生成编码后的三元组文件。

Step5　将 Step4 的结果文件作为此步骤的输入，根据 Step3 中的字典表，可以反转生成原始 RDF 数据文件。

SCOM 算法的总体框架如图 3 所示。

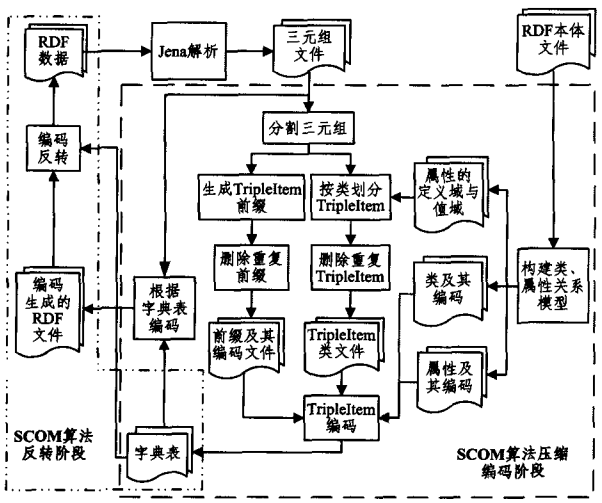

图 3　SCOM 算法总体框架

3.1 SCOM 编码算法

首先使用 Jena(Jena Semantic Web Framework)作为 RDF 数据解析工具，将 RDF XML 格式解析成三元组格式的文件 Fn，然后对 Fn 中的所有的三元组的 TripleItem 按类划分与去重处理，生成多个类文件；对类文件中的 TripleItem 进行编码，生成 TripleItem 与编码的映射字典表；最后，根据字典表和输入的三元组数据完成三元组的编码。

3.1.1 构建类关系和属性关系模型

为了使 RDF 数据的编码具有语义信息，需要生成类编码。由于 RDF 数据中的谓语在本体文件中都有定义，且数量远远少于 RDF 数据中的主语或宾语，因此本阶段在对类进行编码后，需要完成属性的编码。首先将 RDF 数据格式的本体文件进行 Jena 解析，根据类关系生成关系树(子类与父类)，构建类关系的模型。此关系树为一棵多叉树，通过广度优先算法(Breadth-First-Search)[12]并结合定义 4，生成类编码。

以 LUBM(Lehigh University Benchmark)数据集[13]中的类片段为例，假设根据定义 3 所确定的编码位数为 2，则构建的类关系模型如图 4 所示。其中编码的第一位表示类标记，第二位表示直系父类个数标记，第三位和第四位的组合构成了当前类的直系父类节点顺序编码，最后两位构成了当前类的节点顺序编码。图中，Things 类为所有类的父类，它的直系父类个数标记为 0(即 $IPF=0$)。

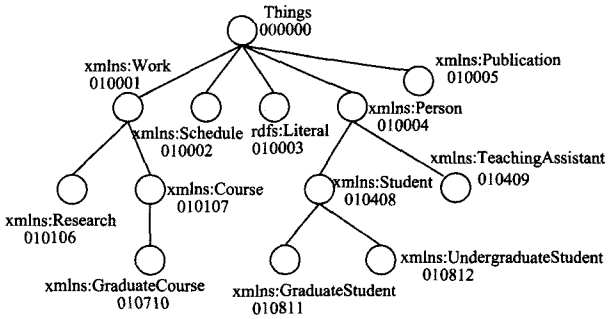

图 4　LUBM 中的部分类关系模型

考虑到一个类的父类可能不止一个，假设 Part-time-GraduateStudent 类(在职研究生)的直系父类为图 4 类关系模型中的 GraduateStudent 类(研究生)和 TeachingAssistant 类(助理)，则 Part-timeGraduateStudent 类的父类节点顺序编码为 GraduateStudent 类和 TeachingAssistant 类的顺序编码的组合(即 0911)。此时，Part-timeGraduateStudent 类的编码为 02091113，其中，$IPF=2$ 表示 Part-timeGraduateStudent 类有两个直系父类。

与构建类关系树类似，构建出属性的关系树，对属性进行编码，与类编码的不同在于属性编码需要添加类编码信息，使得属性编码含有语义信息。

以 LUBM 数据集中的属性片段为例，假设根据定义 3 所确定的编码位数为 2，结合图 4 中的类编码，则属性关系模型如图 5 所示。其中编码的第一位表示属性标记，第二位至第七位的组合为当前属性的类编码(属性的定义域类)，第八位和第九位的组合为当前属性的直系父属性节点顺序编码，最后两位为当前属性的节点顺序编码。此外，本阶段生成了属性定义域与值域文件。

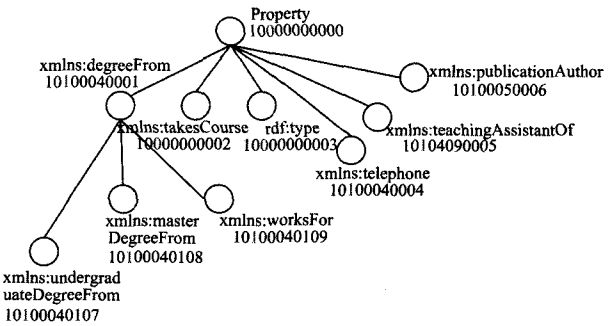

图 5　LUBM 中的部分属性关系模型

采用这种编码方式能够为下文中 RDF 数据的编码新增语义信息。当 TripleItem 为主语或者宾语时，能够根据其编码判断所属类信息；当 TripleItem 为谓语时，能够获取当前谓语的父属性或所属类的信息。

3.1.2 三元组项分类与过滤

SCOM算法中需要将TripleItem按类划分。由于三元组的谓语为本体中的属性，在构建属性关系模型中已经生成了属性编码，因此只需要将三元组的主语和宾语按类划分。由于RDF数据中三元组项可能不唯一，因此在对TripleItem按类划分的同时对TripleItem进行过滤，从而删除重复的TripleItem，确保TripleItem的唯一性，保证相同的TripleItem不会分配到不同的编码。此外，由于不同的TripleItem可能共享相同的URI，为了确保编码具有语义相似性，使得相似的URI编码成相似的数字，根据RDF数据文件抽取出相同的前缀(命名空间)。此外，本阶段需要重写MapReduce的MultipleOutputFormat，使得输出的文件能够按类文件输出。三元组项分类与过滤算法的具体步骤如下。

算法1 三元组项分类与过滤算法

输入：RDF三元组格式文件

输出：TripleItem按类划分的类文件；前缀编码关系文件

```
1. //获取分布式缓存中属性的定义域与值域文件，生成属性与定义域关系的 pDomainMap 和属性与值域关系的 pRangeMap
2. DistributedCache. getLocalCacheFiles(c. getConfiguration());
3. map(key, value)
4.   //key：linenumber (irrelevant)；value：triple；filename：triple 文件名称
5.   for each triple do
6.     //获取当前 triple 的 TripleItem
7.     if triple. predicate=rdf:type then
8.       emit (triple. subject, triple. object)
9.     else
10.      if (class_n=pRangeMap. get(triple. predicate))!=null then
11.        emit (triple. object, class_n)
12.      else if (class_n=pDomainMap. get(triple. predicate))!=null then
13.        emit (triple. object, class_n)
14.      else if triple. object is literal then
15.        emit (triple. object, "Literal")
16.      else
17.        emit(triple. object, "NoneClass");//NoneClass 其他类
18.  for each TripleItem do
19.    if TripleItem not literal then
20.      //根据 TripleItem 的 URI 获取其前缀 prefix
21.      emit (prefix, "prefix");
22.
23. reduce (key, iterator values)
24.   for val in values do
25.     //根据 values 判断其对应的 key 的类型：TripleItem 或前缀
26.     if 当前 key 为前缀 then
27.       //根据重写的 Hash 函数对前缀进行 Hash 编码，设为 prefixID
28.       emit (newKey, key+prefixID) break;
29.   else //TripleItem
30.     emit (val, key) break;
```

3.1.3 生成三元组项编码，构建字典表

获取算法1的结果文件，并将其作为三元组项编码的输入文件。在Map阶段对TripleItem的类文件进行处理，在Reduce阶段对TripleItem进行编码，同时生成TripleItem与其编码的字典映射表文件，将字典映射表文件存储到集群的HDFS上。每个TripleItem编码格式为：所属类编码+前缀编码+尾数编码。TripleItem编码算法的具体步骤如下。

算法2 TripleItem编码算法

输入：TripleItem按类划分的类文件；前缀编码关系文件

输出：字典映射表文件

```
1. //获取内存中的类及其编码；属性及其编码
2. map (key, value)
3.   //key：linenumber (irrelevant)；value：TripleItem
4.   //根据 TripleItem 类文件获取当前类的编码 dictClass
5.   for each TripleItem do
6.     emit (dictClass, TripleItem)
7.
8. reduce (key, iterator values)
9.   localcount=0;//TripleItem 的尾数编码
10.  for val in values do
11.    if val is not dictionary key then //dictPrefix 前缀编码
12.      emit (null, val+","+key+dictPrefix+localCount)
```

TripleItem编码算法生成了TripleItem与其编码的字典映射表，这样能够快速获取TripleItem所对应的编码，提高三元组编码的效率。

以LUBM数据集中的三元组片段(见表1)为例，描述TripleItem编码算法的过程。

表1 输入的RDF三元组数据片段

1. http://www. Department0. University0. edu/UndergraduateStudent99, http://swat. cse. lehigh. edu/onto/univ-bench. owl#takesCourse, http://www. Department0. University0. edu/Course1
2. http://www. Department0. University0. edu/UndergraduateStudent99, http://swat. cse. lehigh. edu/onto/univ-bench. owl#takesCourse, http://www. Department0. University0. edu/Course43
3. http://www. Department0. University0. edu/GraduateStudent93, http://swat. cse. lehigh. edu/onto/univ-bench. owl#teachingAssistantOf, http://www. Department0. University0. edu/Course1
4. http://www. Department0. University0. edu/GraduateCourse17, http://www. w3. org/1999/02/22-rdf-syntax-ns#type, http://swat. cse. lehigh. edu/onto/univ-bench. owl#GraduateCourse
5. http://www. Department0. University0. edu/AssociateProfessor2/Publication0, http://swat. cse. lehigh. edu/onto/univ-bench. owl#publicationAuthor, http://www. Department0. University0. edu/GraduateStudent93
6. http://www. Department2. University0. edu/UndergraduateStudent0, http://swat. cse. lehigh. edu/onto/univ-bench. owl#telephone, xxx-xxx-xxxx

为了下文描述的简洁性，根据表1生成TripleItem编号和原始数据的映射关系表，如表2所列。其中，表1中第3个三元组的主语与第5个三元组的宾语相同，因此只对应一个TripleItem。

将表1所列的三元组片段作为算法1的输入，得到TripleItem的类文件与前缀编码关系文件。假设获取的前缀编码关系文件如表3所列。

将算法1的结果文件作为TripleItem编码算法(算法2)的输入，并且结合文中3.1.1节中的类(属性)关系模型可获取TripleItem的编码。假设阈值α表示TripleItem编码中的尾数位数，若$\alpha=3$，根据表1的三元组片段，结合表2和表3所得到的TripleItem编码信息如表4所列。

表 2　TripleItem 编号和原始数据的映射关系

TripleItem	TripleItem 编号
http://www. Department0. University0. edu/UndergraduateStudent99	TIt1
http://swat. cse. lehigh. edu/onto/univ-bench. owl # takesCourse	TIt2
http://www. Department0. University0. edu/Course1	TIt3
http://www. Department0. University0. edu/Course43	TIt4
http://www. Department0. University0. edu/GraduateStudent93	TIt5
http://swat. cse. lehigh. edu/onto/univ-bench. owl # teachingAssistantOf	TIt6
http://www. Department0. University0. edu/GraduateCourse17	TIt7
http://www. w3. org/1999/02/22-rdf-syntax-ns # type	TIt8
http://swat. cse. lehigh. edu/onto/univ-bench. owl # GraduateCourse	TIt9
http://www. Department0. University0. edu/AssociateProfessor2/Publication0	TIt10
http://swat. cse. lehigh. edu/onto/univ-bench. owl # publicationAuthor	TIt11
http://www. Department2. University0. edu/UndergraduateStudent0	TIt12
http://swat. cse. lehigh. edu/onto/univ-bench. owl # telephone	TIt13
xxx-xxx-xxxx	TIt14

表 3　表 1 所列三元组片段的前缀编码信息

前缀	前缀编码
(xmlns:) http://swat. cse. lehigh. edu/onto/univ-bench. owl #	01
(rdf:) http://www. w3. org/1999/02/22-rdf-syntax-ns #	02
http://www. Department0. University0. edu/	03
http://www. Department2. University0. edu/	04
无前缀(例如 Literal 类型数据)	00

表 4　TripleItem 编码信息

TripleItem	TripleItem 编码	前缀编码	所属类编码	备注
TIt1	01081203001	03	010812	
TIt2	10000000002	—	000000	属性
TIt3	01010703001	03	010107	
TIt4	01010703002	03	010107	
TIt5	01081103001	03	010811	
TIt6	10104090005	—	010409	属性
TIt7	01071003001	03	010710	
TIt8	10000000003	—	000000	属性
TIt9	010710	—	—	类
TIt10	01000503001	03	010005	
TIt11	10100050006	—	010005	属性
TIt12	01081204002	04	010812	
TIt13	10100040004	—	010004	属性
TIt14	01000300001	00	010003	

3.1.4　三元组编码

根据算法 2 中生成的字典映射表，对输入的 RDF 三元组格式文件中的每一个三元组进行编码。将 TripleItem 与字典映射表进行连接，从而生成三元组的编码。三元组编码算法的具体步骤如下。

算法 3　三元组编码算法

输入：RDF 三元组格式文件；字典映射表文件

输出：编码生成的 RDF 文件

```
1. //加载字典表到内存
2. map(key,value)
3.   //key:linenumber (irrelevant);value:triple;filename:triple 文件名称
4.   for each triple do
5.     //获取当前 triple 的 TripleItem
6.     for each TripleItem do
7.     //根据字典表获取 TripleItem 的编码
8.     triple. subject→sID;triple. predicate→pID;triple. object→oID
9.     tripleID=sID+pID+oID;//生成编码后的三元组
10.     emit (filename,tripleID)
11.
12. reduce(key,iterator values)
13. for val in values do
14.   emit (key,val)
```

3.2　SCOM 反转算法

SCOM 算法是一个无损压缩算法。SCOM 中的反转算法能够快速地、完全地恢复编码后的 RDF 数据文件为原始数据。由于 SCOM 算法建立了 TripleItem 及其编码的字典表，结合字典表，可以很容易地将编码生成的 RDF 文件反转成原始的 RDF 文件。为了更加明确 SCOM 反转算法，以伪代码的形式将其描述如下。

算法 4　SCOM 反转算法

输入：编码生成的 RDF 文件

输出：原始的 RDF 三元组文件

```
1. //加载字典表到内存
2. map(key,value)
3.   //key:linenumber (irrelevant);
4.   //value:编码后的三元组 dictTriple;filename:文件名称
5.   for each dictTriple do
6.       //获取当前 dictTriple 的 TripleItem
7.     for each TripleItem do
8.       //将 TripleItem 与字典表进行连接
9.      dictTriple. subject→s
10.      dictTriple. predicate→p
11.      dictTriple. predicate→o
12.      triple=s+p+o;//重构三元组
13.      emit (filename,triple)
14.
15. reduce (key,iterator values)
16. for val in values do
17.      emit (key,val)
```

4　实验分析

实验所使用的硬件环境为 Intel (R) Core(TM) i5-3570，3.40GHz 双核双线程 CPU，2GB 内存，500GB 磁盘，7200rpm 转速。软件环境为 Linux Ubuntu 操作系统，采用 Java 作为编程语言，开发环境为 Eclipse。采用表 5 所列集群工作站的配置作为本系统 Hadoop 集群的配置，共计 4 台，其中 1 台作为 HDFS 的名称节点兼 MapReduce 的主节点，3 台作为 HDFS 的数据节点兼 MapReduce 的从节点。

表 5　集群工作站的基本配置

CPU	Intel(R) Core(TM) i5-3570M
内存	2G
硬盘	500 GB SATA
Java	JDK1.6
Hadoop	Hadoop 1.0.4

本文采用 LUBM(Lehigh University Benchmark)数据集[13]和 DBpedia 数据集[2]对 SCOM 算法行测试。数据集的

基本参数说明如表 6 所列。

表 6　数据集的基本参数说明

数据集名称	三元组数（万个）	三元组文件大小(GB)	属性数（个）	类数（个）
LUBM50	686	1.20	51	43
LUBM100	1382	2.40	51	43
DB3.7	1433	1.90	1662	336
DB3.8	1696	2.30	1794	376
DB3.9	2188	3.00	2352	570

文献[5-8]为目前 RDF 数据分布式并行压缩的最新研究成果，其中，文献[5]所提算法适用于专业的计算机，文献[6,7]中的算法采用 X10 对 RDF 数据进行分布式压缩，文献[8]为目前最先进最成熟的 MapReduce 分布式 RDF 数据压缩算法。因此，本文基于 Hadoop 环境就 SCOM 算法与文献[8]中提出的 MapReduce 压缩算法在不同数据集下对 RDF 数据的压缩进行对比分析。执行压缩编码和反转时，在每个数据集上运行 5 次并取平均响应时间，不同数据集下两种算法的压缩编码情况对比如表 7 所列。不同数据集下两种算法的压缩和反转阶段的时间对比如图 6 所示。

表 7　不同数据集下两种算法的压缩编码情况对比

数据集名称	输入文件（GB）	SCOM 压缩文件（GB）	MapReduce 压缩文件（GB）	SCOM 编码时间（s）	MapReduce 编码时间（s）
LUBM50	1.20	0.29	0.14	368.678	436.597
LUBM100	2.40	0.60	0.27	733.471	930.837
DB3.7	1.90	0.51	0.29	530.358	900.768
DB3.8	2.30	0.60	0.35	639.095	967.704
DB3.9	3.00	0.81	0.46	987.028	1235.216

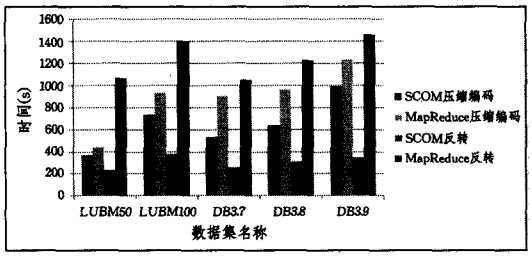

图 6　不同数据集下两种算法的压缩和反转阶段的时间对比

从表 7 和图 6 可知，不同数据集下 SCOM 算法在压缩编码阶段和反转阶段均优于 MapReduce 压缩算法。主要由于 MapReduce 压缩算法没有对 TripleItem 进行过滤，因此在对 TripleItem 编码时，需不断地访问字典表以判断 TripleItem 是否已经编码，造成不必要的效率损失；SCOM 算法由于对 TripleItem 进行分类过滤，剔除了重复的 TripleItem，因此可以直接完成 TripleItem 的编码，提高了编码效率。在反转阶段，SCOM 算法只需要启动一个 MapReduce 任务即可完成反转，而 MapReduce 压缩算法需要启动 4 个 MapReduce 任务才能够完成反转阶段。但是 SCOM 算法的压缩比例低于 MapReduce 压缩算法，源于 SCOM 算法中的编码增加了类信息与前缀信息，类编码和前缀编码占用了一定的文件大小。

为了测试 SCOM 算法在单位时间内压缩或反转所传送的数据量，在不同数据集下对比了两种算法吞吐量，如图 7 所示。从图 7 可知，SCOM 算法的吞吐量均优于 MapReduce 压缩算法，尤其在反转阶段的优势更加明显。SCOM 算法的反转阶段优于压缩编码阶段，每秒最高处理的三元组数达到了 63122 个。

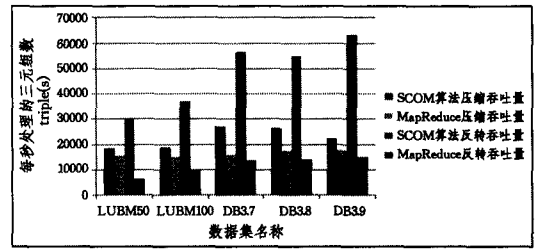

图 7　不同数据集下两种算法吞吐量的对比

由于输入的 RDF 三元组数据分割成 TripleItem 时会产生大量重复的数据，重复的数据使得在对 TripleItem 进行编码时需要不断地访问字典表，以判断当前的 TripleItem 是否已经编码，增加了内存的消耗且浪费时间，同时重复的 TripleItem 在一定程度上会造成系统资源无谓的浪费并增加网络的开销。文中 3.1.2 节提出的三元组项的分类与过滤算法能够减少大量的重复 TripleItem。为了评估算法的有效性，将过滤前后的 TripleItem 进行对比，如表 8 所列。过滤后 TripleItem 的数量远远少于过滤前的 TripleItem 数量，在所测试的数据集范围内，重复 TripleItem(谓语 P)最高达到了过滤前数量的 99%；重复 TripleItem(主语 & 宾语 SO)最高达到了过滤前数量的 94%。可见，MapReduce 压缩算法没有对 TripleItem 进行过滤，增加了系统无谓的开销。

表 8　SCOM 算法在不同数据集下三元组项过滤前后的对比

数据集名称	过滤前 TripleItem		过滤后 TripleItem	
	主语 & 宾语(SO)个数	谓语(P)个数	主语 & 宾语(SO)个数	谓语(P)个数
LUBM50	13726454	6863227	1639678	17
LUBM100	27648874	13824437	3301701	17
DB3.7	28675548	14337774	1688404	1281
DB3.8	33934408	16967204	1981297	1300
DB3.9	43766554	21883277	2835870	1359

此外，为了更加明确 SCOM 算法在各阶段的运行情况，在不同数据集下对比了 SCOM 算法各阶段的时间，如图 8 所示。

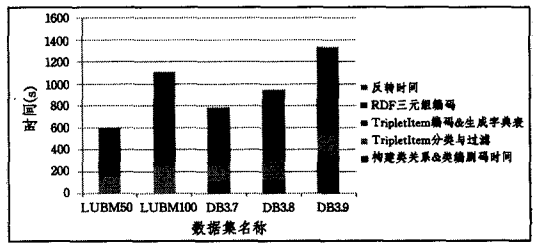

图 8　SCOM 算法在不同数据集下各阶段的时间对比

结束语　本文提出的 SCOM 算法能够在大规模数据下高效地完成 RDF 数据的分布式并行语义编码，并能够实现 RDF 数据的反转。SCOM 算法的编码方式使得 RDF 数据带有语义信息，利于分布式的查询与推理，但字典表的加载受限于集群的内存大小。下一步将会在此方面进行改进，且将算法应用于分布式的查询和推理之中。

参 考 文 献

[1] Du Fang, Chen Yue-guo, Du Xiao-yong. RDF Query Processing Techniques[J]. Journal of Software, 2013, 24(6): 1222-1242(in Chinese)

（下转第 212 页）

表 5 Adult 数据集实验结果

算法	抽样集成算法				改进的
方案	一	二	三	四	K-prototypes
耗时(s)	2.508	2.794	2.814	2.827	12.813
ARI	0.412	0.412	0.435	0.407	0.410
CUM	0.211	0.206	0.223	0.208	0.190

表 6 Covertype 数据集实验结果

算法	抽样集成算法				改进的
方案	一	二	三	四	K-prototypes
耗时(s)	936.6	947.1	989.2	1013	2766
ARI	0.168	0.182	0.179	0.189	0.172
CUM	2.298	2.373	2.394	2.407	2.360

表 7 Poker Hand 数据集实验结果

算法	抽样集成算法				改进的
方案	一	二	三	四	K-prototypes
耗时(s)	1117	1283	1475	1733	6034
ARI	0.120	0.126	0.119	0.128	0.114
CUM	1.363	1.395	1.351	1.369	1.357

实验表明，与改进的 K-prototypes 算法相比，在有效性指标基本相同的情况下，算法所耗时间大大减少，证明该算法对大规模混合数据聚类是有效的。

结束语 本文针对 K-prototypes 算法以及改进的 K-prototypes 算法在对大规模混合数据进行聚类时时间效率不高的问题，提出了一种基于抽样的大规模混合数据集成聚类算法，并在 UCI 数据集中选取的 3 个不同规模的数据集上分别进行了实验。实验表明，该算法在时间效率和聚类精度上与原有算法相比有较大提升，证明该算法是有效的。

参 考 文 献

[1] MacQueen J B. Some methods for classification and analysis of multivariate observations [C] // Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley: University of California, 1967: 281-297

[2] Ruspini E R. A new Approach to clustering [J]. Information and Control, 1969, 15(1): 22-32

[3] Camastra F, Verri A. A novel kernel method for clustering [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(5): 801-805

[4] Zhang T, Ramakrishnan R, Livny M. BIRCH [C] // Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. Quebec: ACM, 1996: 103-114

[5] Guha S, Rastogi R, Shim K. CURE: An efficient clustering algorithm for clustering large databases [C] // Proceedings of the Symposium on Management of Data (SIGMOD). Seattle: ACM, 1998: 73-84

[6] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters inlarge spatial databases with noise [C] // Proceedings of the 2th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. USA: AAAI, 1996: 226-231

[7] Huang Zhe-xue. Extensions to the k-means algorithm for clustering large data sets with categorical values [J]. Data Mining and Knowledge Discovery, 1998, 2(3): 283-304

[8] Liang Ji-ye, Zhao Xing-wang, Li De-yu, et al. Determining the number of clusters using information entropy for mixed data [J]. Pattern Recognition, 2012, 45(6): 2251-2265

[9] He Zeng-you, Xu Xiao-fei, Deng Sheng-chun. Clustering Mixed Numeric and Categorical Data: A Cluster Ensemble Approach [J]. Computer Science Artificial Intelligence, 2005, 5(4): 225-268

[10] Luo Hui-lan, Wei Hui. Clustering Algorithm for Mixed Data Based on Clustering Ensemble Technique [J]. Computer Science, 2010, 37(11): 234-238(in Chinese)
罗慧兰，危辉. 一种基于聚类集成技术的混合型数据聚类方法 [J]. 计算机科学，2010，37(11)：234-238

[11] Zhou Zhi-hua, Tang Wei. Clusterer ensemble [J]. Knowledge-Based Systems, 2006, 19(1): 77-83

[12] Yang Cao-yuan, Liu Da-you, Yang Bo, et al. Research on Cluster Aggregation Approaches [J]. Computer Science, 2011, 38(2): 166-170(in Chinese)
杨草原，刘大有，杨博，等. 聚类集成方法研究 [J]. 计算机科学，2011，38(2)：166-170

(上接第 202 页)

杜方，陈跃国，杜小勇. RDF 数据查询处理技术综述 [J]. 软件学报，2013，24(6)：1222-1242

[2] Auer S, Bizer C, Kobilarov G, et al. Dbpedia: A nucleus for a web of open data [M]. Springer Berlin Heidelberg, 2007: 722-735

[3] Apweiler R, Bairoch A, Wu C H, et al. UniProt: the universal protein knowledgebase [J]. Nucleic Acids Research, 2004, 32 (suppl 1): D115-D119

[4] Stadler C, Lehmann J, Höffner K, et al. Linkedgeodata: A core for a web of spatial open data [J]. Semantic Web, 2012, 3(4): 333-354

[5] Goodman E L, Jimenez E, Mizell D, et al. High-performance computing applied to semantic databases [M] // The Semanic Web: Research and Applications. Springer Berlin Heidelberg, 2011: 31-45

[6] Long Cheng, Malik A, Kotoulas S, et al. Efficient parallel dictionary encoding for RDF data [C] // Proceedings of the 17th International Workshop on the Web and Databases (WebDB). 2014

[7] Long Cheng, Malik A, Kotoulas S, et al. Scalable RDF Data Compression using X10 [J/OL]. http://rian.ie/ga/item/viem/109810.html

[8] Urbani J, Maassen J, Drost N, et al. Scalable RDF data compression with MapReduce [J]. Concurrency and Computation: Practice and Experience, 2013, 25(1): 24-39

[9] Wu Bu-wen, Jin Hai, Yuan Ping-peng. Scalable SAPRQL querying processing on large RDF data in cloud computing environment [M] // Pervasive Computing and the Networked World. Springer Berlin Heidelberg, 2013: 631-646

[10] Liu Liu, Yin Jiang-tao, Gao Li-xin. Efficient social network data query processing on mapreduce [C] // Proceedings of the 5th ACM workshop on HotPlanet. ACM, 2013: 27-32

[11] Lee D, Kim J S, Maeng S. Large-scale incremental processing with MapReduce [J]. Future Generation Computer Systems, 2014, 36: 66-79

[12] Thomas H, Cormen Charles E, Leiserson Ronald L, et al. Introduction to Algorithms(第 3 版) [M]. 殷建平，徐云，等译. 北京：机械工业出版社，2013：593-599

[13] Guo Yuan-bo, Pan Zheng-xiang, Heflin J. LUBM: A benchmark for OWL knowledge base systems [J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3(2): 158-182