基于二级索引结构的图压缩算法

李高超[1,2]，李犇[3]，卢毓海[4,5]，刘梦雅[6]，刘燕兵[4,5]

（1. 中国科学院大学网络空间安全学院，北京 100049；2. 国家计算机网络与信息安全管理中心，北京 100029；
3. 北京市公安局朝阳分局，北京 100029；4. 中国科学院信息工程研究所，北京 100093；
5. 信息内容安全技术国家工程实验室，北京 100093；6. 英国南安普顿大学，南安普顿 SO171BJ）

摘 要：目前，各领域对图数据的分析、应用需求日益增加，且对结构复杂、耦合度高的大规模图数据的管理面临着速度低下和空间开销大的双重挑战。面对图数据管理中查询耗时高和空间占比大的难题，提出一种图数据二级索引压缩算法——GComIdx。该算法利用有序的键值（Key-Value）结构将相关节点和边尽可能地以相邻的方式存储，并为高效的属性查询和邻居查询分别构造二级索引和 hash 节点索引。此外，为了节省存储空间，GComIdx 算法采用压缩算法来降低图数据磁盘空间占用率。实验结果表明，GComIdx 算法能够有效降低图数据计算的初始化时间和图数据存储的磁盘空间占用，且查询时间小于通用数据库和其他 Key-Value 存储方案。

关键词：二级索引；图压缩；键值结构；属性查询；邻居查询

中图分类号：TN925

文献标识码：A

doi: 10.11959/j.issn.1000-436x.2018104

Graph compression algorithm based on a two-level index structure

LI Gaochao[1,2], LI Ben[3], LU Yuhai[4,5], LIU Mengya[6], LIU Yanbing[4,5]

1. School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
2. Management Center of National Computer Network and Information Security, Beijing 100029, China
3. Chaoyang Branch of Beijing Public Security Bureau, Beijing 100029, China
4. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
5. National Engineering Laboratory for Information Security Technologies, Beijing 100093, China
6. University of Southampton, Southampton SO171BJ, United Kingdom

Abstract: The demand for the analysis and application of graph data in various fields is increasing day by day. The management of large-scale graph data with complicated structure and high degree of coupling faces two challenges: one is querying speed too slow, the other is space consumption too large. Facing the problems of long query time and large space occupation in graph data management, a two-level index compression algorithm named GComIdx for graph data was proposed. GComIdx algorithm used the ordered Key-Value structure to store the associated nodes and edges as closely as possible, and constructed two-level index and hash node index for efficient attribute query and neighbor query. Furthermore, GComIdx algorithm used a graph data compressed technology to compress the graph data before it directly stored in hard disk, which could effectively reduce the storing space consumption. The experimental results show that GComIdx algorithm can effectively reduce the initialization time of the graph data calculation and the disk space occupancy of the graph data storing, meanwhile, the query time is less than common graph databases and other Key-Value storage solutions.

Key words: two-level index, graph compress, Key-Value structure, attribute query, neighbors query

收稿日期：2017-11-08；**修回日期**：2018-03-30

通信作者：卢毓海，luyuhai@iie.ac.cn

基金项目：国家重点研发计划基金资助项目（No.2016YFB0800300）；中国科学院信息工程研究所基础前沿基金资助项目（No.Y7Z0351101）

Foundation Items: The National Key R&D Program of China (No.2016YFB0800300), The Fundamental Theory and Cutting Edge Technology Research Program of Institute of Information Engineering, CAS (No.Y7Z0351101)

1 引言

随着网络中大规模复杂多变数据的产生，数据分析的关注点从实体属性逐渐转向实体间的关系[1]，图成为社交网络、信息检索等领域研究分析数据的有效模型[2-4]，各种面向实体间关系的基于图数据的应用层出不穷。大规模图数据结构复杂、耦合度高等特点使查询操作时间开销大、缓存命中率低，并给数据管理带来了巨大的挑战，导致当前业界应用广泛的数据库在存储和检索图数据方面显得力不从心[5-6]。优化图数据查询访问算法能够满足图数据存储、计算、更新等操作节省处理时间和存储空间的需求。

在大规模实体关系图中，查询计算分析以节点属性查询、边属性查询和节点邻居查询为主。带有属性信息的大规模图数据通常无法完全加载到内存中，提高节点邻居查询命中缓存的概率能够有效减少查询的时间。本文针对上述图数据查询需求，提出了图数据压缩索引算法——GComIdx。GComIdx 在属性图模型的核心思想基础上，分别对属性查询和节点邻居查询这 2 个方面进行了优化，在超大规模简单图的查询方面有着较好的表现，且 GComIdx 主要针对的是静态图的应用场景。

2 二级索引图压缩算法

本节提出针对实体关系网络查询需求的图数据压缩索引算法——GComIdx，该算法支持的查询操作有节点属性查询、边属性查询和节点邻居查询。以图 1 中的图数据为例逐一说明该算法的工作过程。图 1 为微博业务中常见的用户、文章产生的关系网络，其中，节点包括用户和文章 2 种，关系包括关注、互粉、发布、转发、收藏等。

2.1 属性查询

在属性查找方面，GComIdx 将节点和边的属性存储模型抽象成 Key-Value 形式，按照 Key 将数据进行排序后压缩存储，当查找节点和边的属性时，采用二分查找的方式实现属性的快速查找。

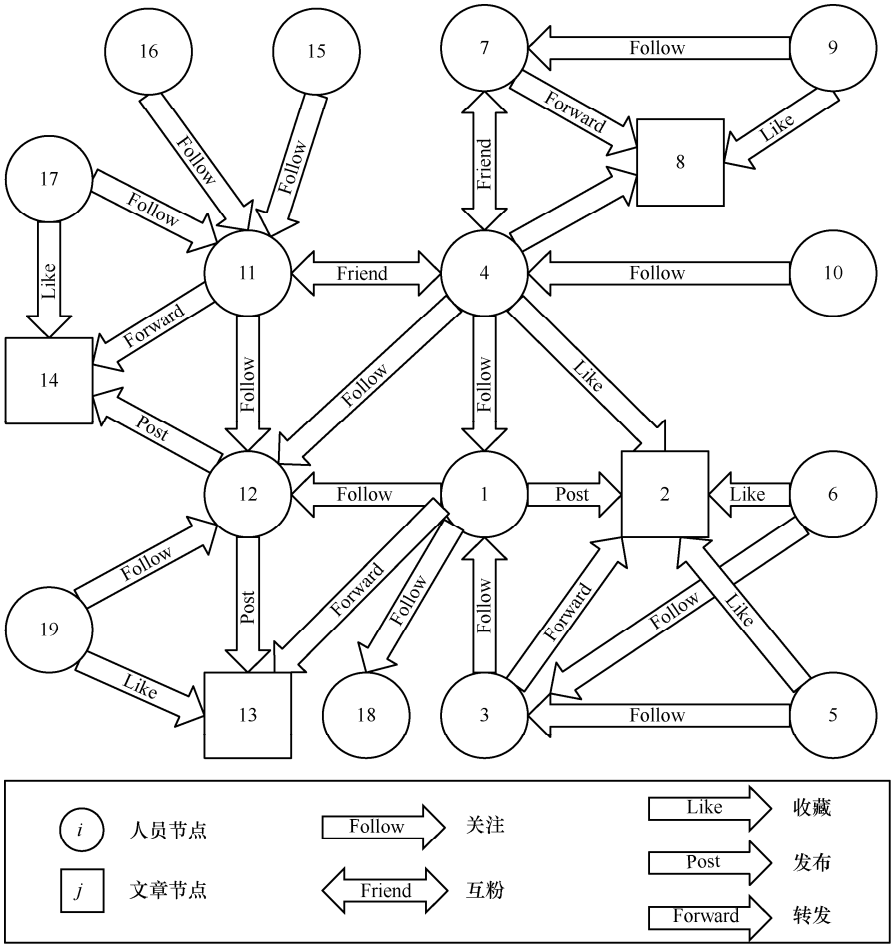


图 1 微博关系数据

2.1.1 属性压缩索引构建

GComIdx 算法中属性压缩索引构建过程主要包括以下 3 个步骤：排序、分块、压缩。最终，该算法在解决属性图模型中属性查找的问题方面，采用在有序的 Key-Value 数据集上建立二级压缩索引的方式实现。

首先，本文算法将节点和边的数据抽象为 Key-Value[7]形式。其中，Key 为<u_0,v_0>的形式，Value 为按照特定规则处理的序列化数据。

在节点数据抽象方面，本文算法将节点进行编号，分配节点 *ID*，节点 *ID* 从 1 到 *n* 编号。第 *i* 个节点属性数据抽象为 Key-Value，即<*i*,0>-Attr^{i}，其中，Key 为<*i*,0>（u_0 为 *i*，v_0 为 0），Value 为本节点的属性数据 Attr^{i}。

在边属性数据抽象方面，本文算法将节点 *u* 与节点 *v* 之间的关系 Attr$^{<u,v>}$抽象为 Key-Value 形式，即<*u*,*v*>- Attr$^{<u,v>}$，其中，Key 为<*u*,*v*>（u_0 为 *u*，v_0 为 *v*），Value 为节点 *u* 和节点 *v* 之间的关系属性数据 Attr$^{<u,v>}$。本文算法对原始 Key 进行处理，将 u_0 和 v_0 的 *ID* 转换为二进制后，再将 u_0 左移 32 位末尾补 0，其结果与 v_0 做与操作，处理后得到一个 64 位的长整型 Key。

最后，逐条读入节点和边的属性数据并对其按照源节点 *ID* 进行分桶，在桶内以 64 位的长整型 Key 进行排序，同时记录源节点的出度，过程如图 2 所示。

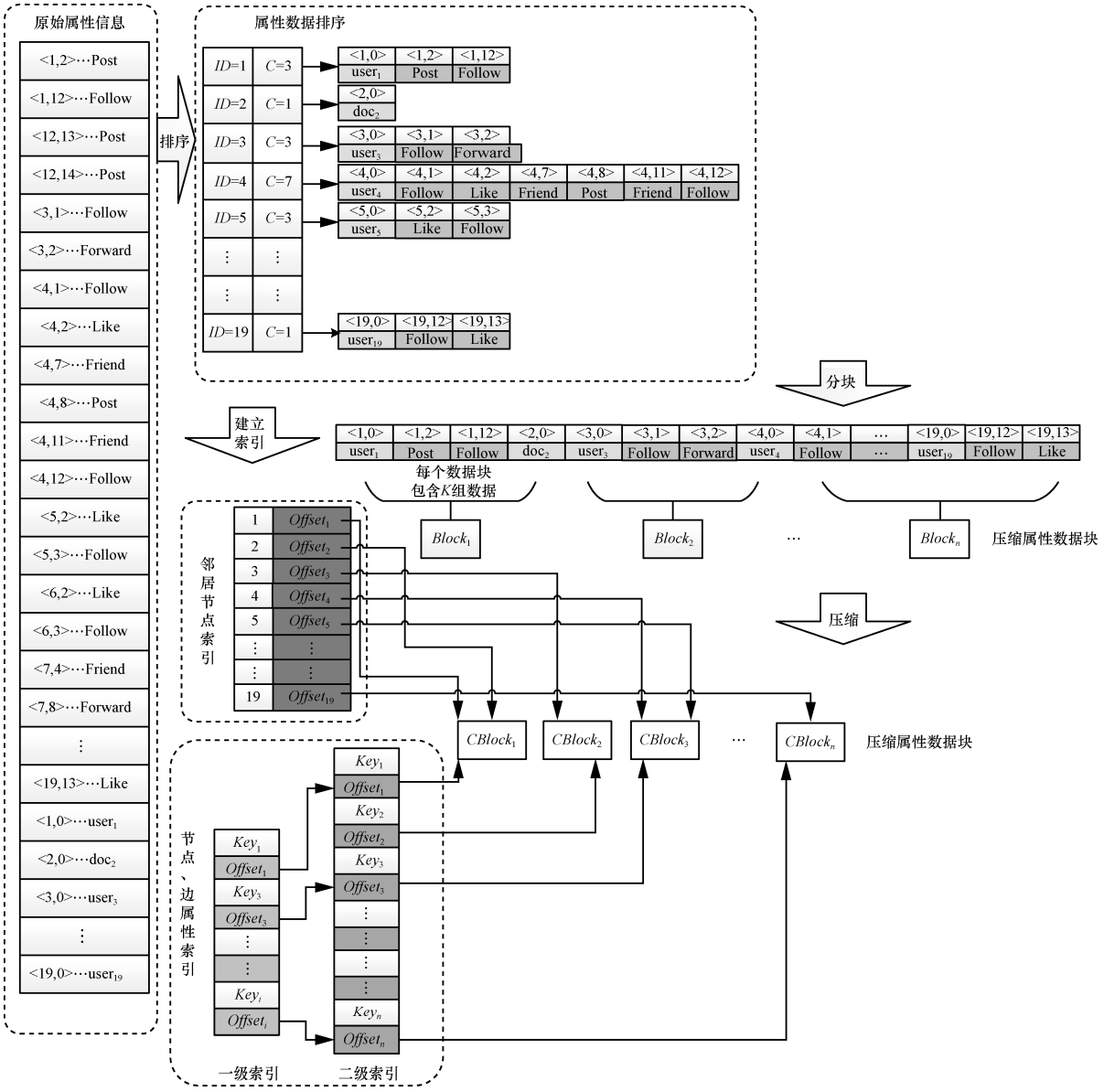


图 2　GComIdx 索引构造过程

图数据对应的有序 Key-Value[7]列表生成方式如图 3 所示，将节点和边的属性存储模型抽象成 Key-Value 数据集，并将数据集按 Key 值进行排序。基于 Key 值有序的属性数据具有以下 3 个方面的优势：1)在查找属性信息时，可以在有序的数据上进行二分查找，提高查找的效率；2)有序的数据具有相似性，当这些数据集中存放在同一个数据块中，在对数据块进行压缩时能获得较好的压缩比[8-9]；3)从当前计算机硬件发展的规律来看，CPU 的计算已不再是系统瓶颈，而是内存空间大小和磁盘 I/O 速度[10]，压缩比较高的数据块载入内存与解压缩时间开销远小于将原始数据读入内存的时间开销。

在已经排序的 Key-Value 数据集上对数据进行分块和压缩，如图 3 所示按照压缩分块的分布情况建立二级索引。首先根据数据集的大小和数据块的大小确定低级索引的数据规模，一般以低级索引能够常驻内存为标准，从而加速查找速度。同理，在低级索引的基础上建立高级索引，高级索引指向低级索引中的相应位置。

GComIdx 属性查询索引的构建如算法 1 所示，算法输入为节点出度统计表 C、有序的边列表文件 E 和数据块规模 s。步骤 1)~步骤 8)根据节点出度和块规模 s 对节点进行分块，得到分块后的每个块所包含的边数目。步骤 9)~步骤 25)根据已经分好的块大小逐行读取边数据信息，对边数据抽象成属性模型并将 Key 和 Value 分别读入内存的同一数据块中，对数据进行压缩，最后将压缩的数据块写入磁盘。

算法 1 GComIdx 属性查询索引构建算法

1) GComIdx-Property (C , E , s)
2) for C 中的每一个元素 do
3) $n \leftarrow n+element.count$
4) if $n > s$ do
5) Push(*element.node*,*B*)
6) Push(*n*,*S*)
7) end if
8) end for
9) $EdgeOffset \leftarrow 0$
10) for B 中的每一个元素 do
11) $Offset \leftarrow 0$
12) while 从 E 中读取一行元素 do
13) *Key*,*Value* ← *line*
14) Push(*Key*,*DBlock*)

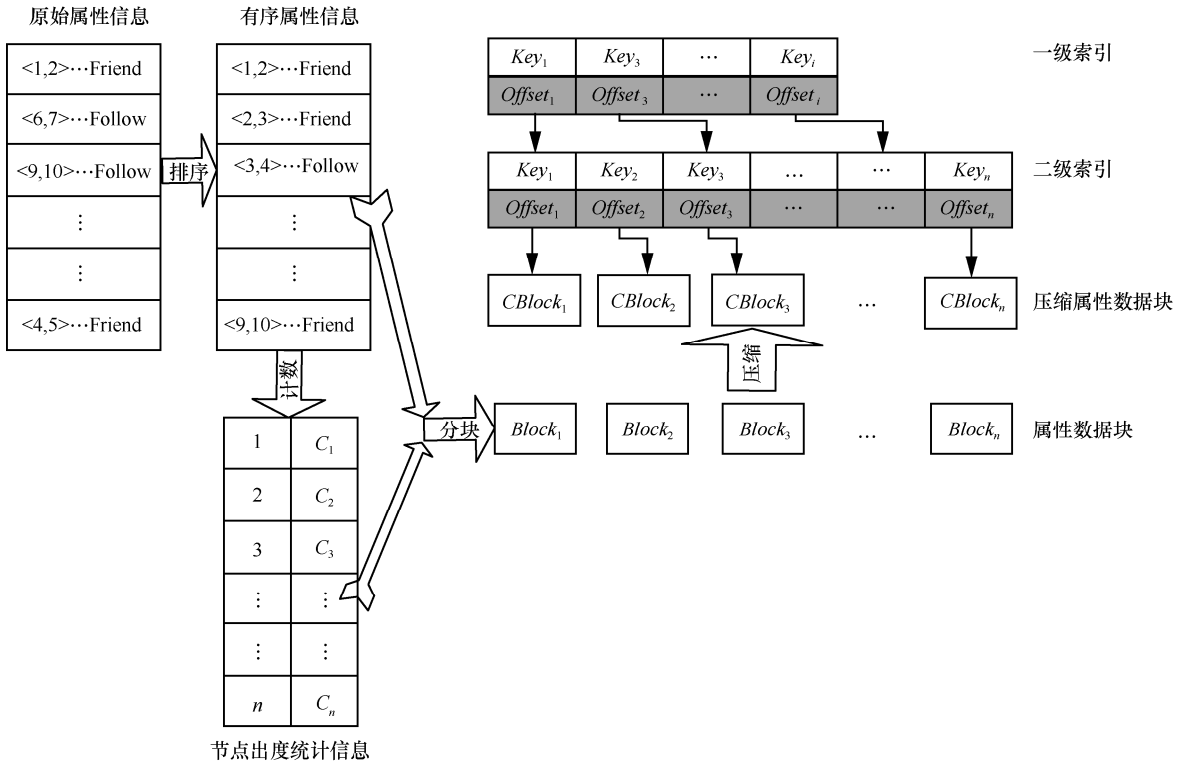


图 3 GComIdx 属性查询索引构造过程

```
15)            Push(Offset,DBlock)
16)            Push(Value,DBlock[Offset])
17)            Offset ← Offset + length(Value)
18)            if Offset == 0 do
19)                Push(Key,Edges_Index)
20)            Push(EdgeOffset,Edges_Offset)
21)            end if
22)        end while
23)        CBlock ← Compress(DBlock)
24)        append CBlock to the index file
25)    end for
```

2.1.2　属性查询操作

属性的查询操作分为节点的属性查询和边的属性查询 2 种。当进行节点属性查询时，查询的 Key 值为节点 *ID*；当进行边的属性查询时，查询的 Key 值为<*u*,*v*>，其中，*u* 为边的起始节点 *ID*，*v* 为边的结束节点 *ID*，GComIdx 将 *u* 和 *v* 的 *ID* 转换为二进制后组合成一个 64 位的长整型 Key。获得查询 Key 值后，GComIdx 在有序的 Key 值数组上进行二分查找。

2.2　相邻节点查询

在相邻节点查找方面，GComIdx 仍采用有序 Key-Value 列表方式存储边数据。大规模实体关系图中节点数远低于边数[11]，所以 GComIdx 对节点的索引采取一级 hash 索引组织管理数据，即将边列表排序后按照边的起始节点 *ID* 进行 hash 的方式构建索引，以达到快速查找指定节点相邻所有节点的目的。

2.2.1　相邻节点压缩索引构建

GComIdx 将边数据<*u*,*v*>存为其相应 Key 值（见 2.1.2 节），并将所有边按 Key 值排序，节点 *u* 的所有边在数据块中将连续存在，如图 4 所示，根据已经计算得到的节点出度表信息，得出每个节点所对应的边数据段在压缩的边属性信息中所对应的位置。按照 2.1.1 节中相同的分块方法划分得到数据块并压缩，构造邻居节点索引。

2.2.2　相邻节点查询操作

查询给定节点的相邻节点时，根据待查询的节点 *u*，从邻居节点索引中获取该节点的首条边在压缩数据块中的位置 Offset_u，将对应的数据块 CBlock_i 调入内存，执行解压缩和遍历、去重等操作，从而获取节点集合 *V*，其中，$V=\{v|<u,v>\in E\}$。

3　实验评估

3.1　实验数据与实验环境

实验数据为特定行业收集到的大规模社交网络图数据，包括 600 万个节点、2.8 亿条边。

实验环境为曙光服务器 A620-G，具体配置如下：AMD Opteron 6128 2 GHz CPU（主频：3.10 GHz 八核），32 GB DDR3 内存，Linux Centos 7（64 位）操作系统。算法代码以 C++实现，用 GCC 4.8.3 编译。

3.2　评价标准

实验主要从图初始化和子图查询[12]这 2 个方面进行，因子图查询需要同时执行属性查询和邻居查询。在图数据集上对 PostgreSQL[13]、SSDB、Redis[14]、Neo4j[15]和 GComIdx 数据存储方案进行比较，主要以图数据导入的时间开销、占用磁盘空间和不同规模图数据上的查询操作所耗费的时间作为评价标准。

3.3　索引构建时间比较

将大规模图数据分别导入 PostgreSQL、SSDB、Redis、Neo4j 和 GComIdx，观察记录程序运行时间和磁盘占用空间。各个存储方案的初始化时间如表 1 所示。在同样的实验条件下用时最长的是图数据库 PostgreSQL。由此可知，Neo4j 和 PostgreSQL 等通用数据库比较适用于图规模逐渐增大的场景，如动态图的处理。而采用 Key-Value 结构的存储方案，如 Redis、SSDB、GComIdx，在数据导入方面有着明显优势。

表 1　　存储方案初始化时间

算法名称	时间消耗/s
PostgreSQL	50 400
SSDB	13 338
Redis	7 956
Neo4j	17 280
GComIdx	8 478

GComIdx 在初始化时间开销上有着较好的表现，其用时仅次于 Redis 内存数据库集群。但是由于 Redis 是内存数据库，当实验数据量过大时，单机上的单进程写入多次失败，虽然集群部署方案可以解决写入失败问题，但是引入了查询失败和多次查询的问题，导致查询效率下降。相较而言，GComIdx 以微小的初始化保证正确初始化和成功查询。

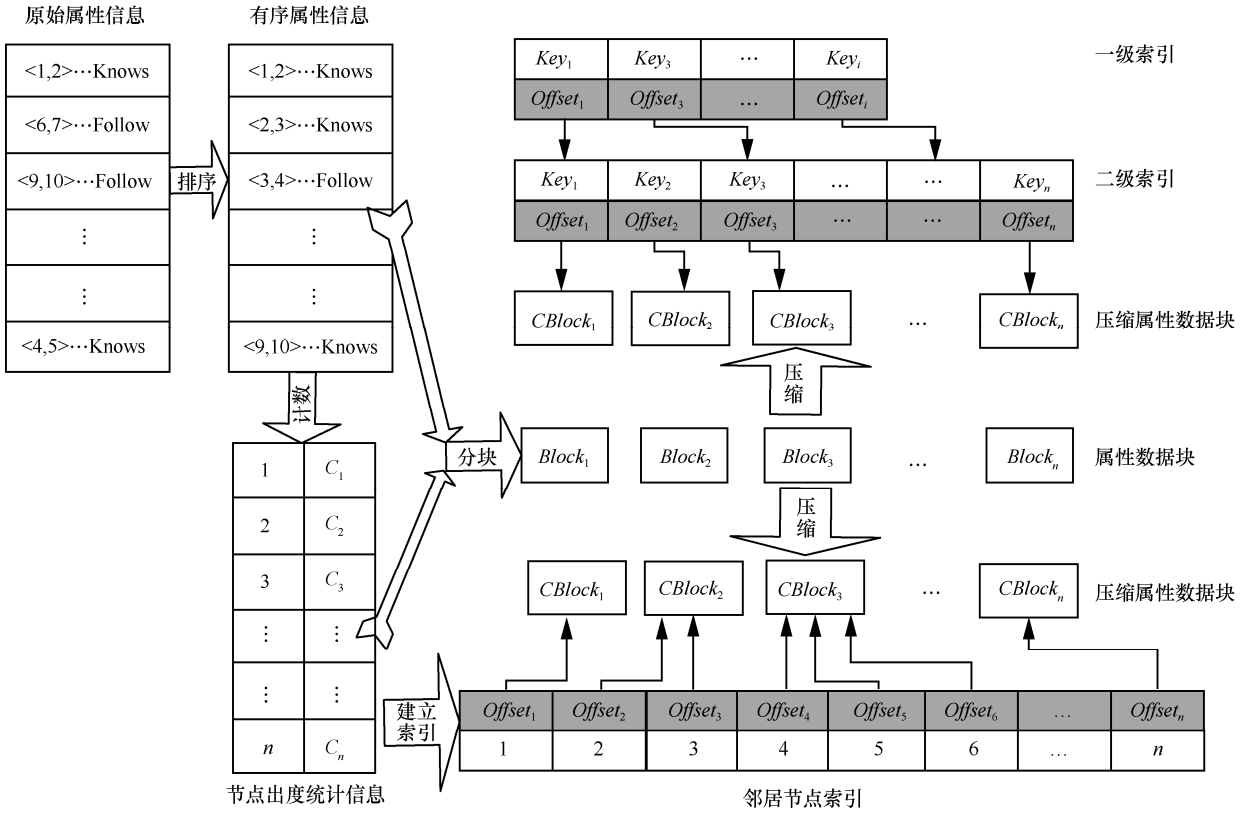


图 4　GComIdx 邻居节点索引构造过程

3.4　索引存储空间比较

实验数据集在各个存储方案初始化处理后占用的磁盘空间如表 2 所示。GComIdx 算法磁盘占用率最低，SSDB、Redis 具有较好的空间性能，Neo4j、PostgreSQL 磁盘占用率较高。Key-Value 结构在节省磁盘空间上同样优于通用数据库，而与其他采用 Key-Value 结构的方案相比，GComIdx 在磁盘占用空间上表现出的优势来源于压缩操作，尽管其生成并存储了二级索引，但其对边序列排序分块后的压缩操作成功降低了磁盘上需要存储的数据量。

表 2　　存储方案磁盘空间

算法名称	磁盘空间占用/GB
PostgreSQL	20.1
SSDB	2.04
Redis	5.05
Neo4j	33.7
GComIdx	1.46

GComIdx 算法充分利用了 Key-Value 结构在初始化和存储空间上的优势，并通过对图数据的排序和分块后的压缩进一步提高了其在节省磁盘空间上的优势。实验说明，无论从初始化时间还是磁盘空间占比来看，GComIdx 都优于其他通用数据库。

3.5　查询效率比较

在导入后的图数据集上，按各个图数据管理方案分别查询边规模为 50、100、500、1 000、5 000、10 000、50 000、100 000 的子图，观察并记录其查询时间的变化，如图 5 所示。从实验结果来看，当查询的子图规模小于 10 000 条边时，各方案查询效率相差不大，查询时间都呈线性增长；当查询的子图规模大于 10 000 条边时，PostgreSQL 与 Neo4j 的性能表现都急剧下降，其他方案表现相差不大。

与其他存储方案相比，GComIdx 算法在子图规模为 50 000 条边以下时，查询耗时最小；当子图规模为 100 000 条边时，其查询耗时与 SSDB 和 Redis 方案近似相同。GComIdx 算法在执行查询操作时，利用二级索引查询得到所需节点或边的信息后，还需要执行数据分块载入内存中的解压缩操作，即实验结果表示的查询时间包含了解压缩操作的时间，因此不难得出，GComIdx 提出的基于 Key 值排序后的二级索引算法在图数据上的查询更具有优势。

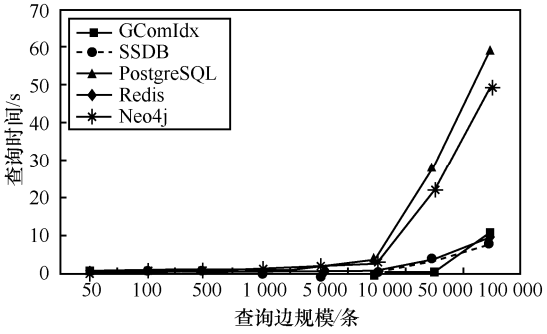


图 5　存储方案查询时间

4　结束语

针对通用数据库在管理图数据时存在的查询耗时高和空间占比大的难题，本文针对图数据上的查询操作，提出二级索引压缩算法——GComIdx。基于有序的 Key-Value 结构，为属性查询和节点邻居查询分别构建二级索引和节点 hash 索引，充分利用了节点相关性提高查询速度，降低数据压缩后的磁盘占用空间。实验结果表明，GComIdx 在有效降低图数据初始化时间开销和磁盘占用空间的情况下，查询效率仍优于通用数据库和其他 Key-Value 存储方案。

参考文献：

[1] 陈忱. 面向 Web 的实体关系查询与分析关键技术研究[D]. 沈阳: 东北大学, 2013.
CHEN C. Study on key techniques of web focused entity relation query processing and analyzing[D]. Shenyang: Northeastern University, 2013.

[2] 程学旗, 靳小龙, 王元卓, 等. 大数据系统和分析技术综述[J]. 软件学报, 2014, 25(9): 1889-1908.
CHENG X Q, JIN X L, WANG Y Z, et al. Survey on big data system and analytic technology[J]. Journal of Software, 2014, 25(9): 1889-1908.

[3] 夏黎明. 云环境下数据关联管理机制的研究及其在铁路行业的应用实现[D]. 北京: 北京交通大学, 2011.
XIA L M. Research for data association management mechanism and its application Implementation in railway[D]. Beijing: Beijing Jiaotong University, 2011.

[4] 周溜溜. 基于图结构的数据挖掘研究及应用[D]. 南京: 南京林业大学, 2013.
ZHOU L L. Research and application of data mining based on graph structure[D]. Nanjing: Nanjing Forestry University, 2013.

[5] ANGLES R, GUTIERRZ C. Survey of graph database models[J]. ACM Computing Surveys, 2008, 40(1):178-187.

[6] 于戈, 谷峪, 鲍玉斌, 等. 云计算环境下的大规模图数据处理技术[J]. 计算机学报, 2011, 34(10): 1753-1767.
YU G, GU Y, BAO Y B, et al. Large scale graph data processing on cloud computing environments[J]. Chinese Journal of Computers, 2011, 34(10): 1753-1767.

[7] 党永兴. 键值数据库存储引擎设计与实现[D]. 武汉: 华中科技大学, 2014.
DANG Y X. Design and implement of storage engine of key value database[D]. Wuhan: Huazhong University of Science and Technology, 2014.

[8] 张宇, 刘燕兵, 熊刚, 等. 图数据表示与压缩技术综述[J]. 软件学报, 2014, 25(9): 1937-1952.
ZHANG Y, LIU Y B, XIONG G, et al. Survey on succinct representation of graph data[J]. Journal of Software, 2014, 25(9): 1937-1952.

[9] ZHANG Y, XIONG G, LIU Y, et al. Delta-K2-tree for compact representation of Web graphs[C]//Asia-Pacific Web Conference. 2014: 270-281.

[10] 吴昌松. ISCSI 双控制器存储系统缓存设计与实现[D]. 济南: 山东大学, 2014.
WU C S. ISCSI based dual-controller storage system cache design and implementation[D]. Ji'nan: Shandong University, 2014.

[11] GONZALEZ J E, LOW Y, GU H, et al. PowerGraph: distributed graph-parallel computation on natural graphs[C]//Usenix Conference on Operating Systems Design and Implementation. 2012: 17-30.

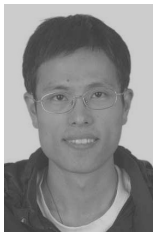
[12] 邹磊. 图数据库中的子图查询算法研究[D]. 武汉:华中科技大学, 2009.
ZOU L. Research on sub graph query algorithm in graph database[D]. Wuhan: Huazhong University of Science and Technology, 2009.

[13] MOMJIAN B. Postgre SQL introduction and concepts[J]. Journal of Conflict Resolution, 2001(3): 353-355.

[14] 郎泓钰, 任永功. 基于 Redis 内存数据库的快速查找算法[J]. 计算机应用与软件, 2016, 33(5): 40-43.
LANG H Y, REN Y G. A fast search algorithm based on Redis memory database[J]. Computer Applications and Software, 2016, 33(5): 40-43.

[15] WEBBER J. A programmatic introduction to Neo4j[C]//Conference on Systems, Programming, and Applications: Software for Humanity. 2012: 217-218.

[作者简介]



李高超（1985-），男，北京人，中国科学院大学博士生，主要研究方向为计算机软件、网络安全、下一代网络、SDN 技术等。

李犇（1987-），男，山东济宁人，北京市公安局朝阳分局副主任科员，主要研究方向图数据库管理技术。

卢毓海（1987-），男，江西赣州人，中国科学院信息工程研究所助理研究员、博士生，主要研究方向为模式串匹配与信息过滤。

刘梦雅（1991-），女，河北石家庄人，英国南安普顿大学博士生，主要研究方向为关联数据、数据整合、数据市场等。

刘燕兵（1981-），男，湖北麻城人，博士，中国科学院信息工程研究所副研究员，主要研究方向为模式串匹配与信息过滤、图数据分析与挖掘等。